UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

IZABELLA CRISTINE OLIVEIRA REZENDE

R2MDD: UM *FRAMEWORK* PARA RASTREABILIDADE E MONITORAMENTO DE REQUISITOS COM FOCO NO DESENVOLVIMENTO DIRIGIDO A MODELOS

SÃO CRISTÓVÃO
2016

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

IZABELLA CRISTINE OLIVEIRA REZENDE

R2MDD: UM *FRAMEWORK* PARA RASTREABILIDADE E MONITORAMENTO DE REQUISITOS COM FOCO NO DESENVOLVIMENTO DIRIGIDO A MODELOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção de título de Mestre em Ciência da Computação.

Orientadora: Profª. Drª. Adicinéia Aparecida de Oliveira

SÃO CRISTÓVÃO

2016

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL UNIVERSIDADE FEDERAL DE SERGIPE

Rezende, Izabella Cristine Oliveira

R467r R2MDD: um framework para rastreabilidade e monitoramento de requisitos com foco no desenvolvimento dirigido a modelos / Izabella Cristine Oliveira Rezende; orientadora Adicinéia Aparecida de Oliveira. – São Cristóvão, 2016.

115f.: il.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Sergipe, 2016.

1. Engenharia de software. 2. Programas de computador. 3. Software - Desenvolvimento. I. Oliveira, Adicinéia Aparecida de. II. Título

CDU: 004.41

IZABELLA CRISTINE OLIVEIRA REZENDE

R2MDD: UM *FRAMEWORK* PARA RASTREABILIDADE E MONITORAMENTO DE REQUISITOS COM FOCO NO DESENVOLVIMENTO DIRIGIDO A MODELOS

BANCA EXAMINADORA

Profª. Drª. Adicinéia Aparecida de Oliveira, Orientadora
Universidade Federal de Sergipe (UFS)

Profº. Dr. Michel dos Santos Soares, Membro Interno
Universidade Federal de Sergipe (UFS)

Profª. Drª. Fernanda Maria Ribeiro Alencar, Membro Externo
Universidade Federal de Pernambuco (UFPE)

R2MDD: UM *FRAMEWORK* PARA RASTREABILIDADE E MONITORAMENTO DE REQUISITOS COM FOCO NO DESENVOLVIMENTO DIRIGIDO A MODELOS

Este exemplar corresponde à redação final da Dissertação de Mestrado, sendo o Exame de Defesa da mestranda Izabella Cristine Oliveira Rezende para ser aprovada pela Banca Examinadora.

São Cristóvão – SE, 18 de agosto de 2016.

_____

Profª. Drª. Adicinéia Aparecida Oliveira
Orientadora

_____

Profº. Dr. Michel dos Santos Soares
Membro Interno

_____

Profª. Drª. Fernanda Maria Ribeiro Alencar
Membro Externo

DEDICATÓRIA

Dedico este trabalho a Deus, que permitiu que tudo pudesse ser realizado, e a minha família, que apoia continuamente os meus estudos, tem paciência e compreende minhas ausências, como também compartilhou comigo os momentos bons e difíceis ocorridos durante a trajetória.

AGRADECIMENTOS

Agradeço a Deus pelas bênçãos sem fim e pela alegria de viver em Tua presença. Senhor, obrigada por ter me dado forças para não desistir e superar todas as dificuldades pelas quais passei durante o período de mestrado.

Aos meus pais, Manoel Laurentino e Maria Célia, por serem meus exemplos, me orientarem e mostrarem que as conquistas só ocorrem através de estudo, aprendizado e muito trabalho, por me incentivarem e apoiarem sempre. Meu infinito agradecimento.

À minha orientadora Adicinéia, por toda paciência, compreensão, ótima orientação e auxílio, bem como por compartilhar seu conhecimento, atenção, amizade e apoio. Obrigada por tudo!

Às minhas irmãs, Caroline e Yanara por sempre acreditarem em mim, se orgulharem e estarem presentes nos momentos que mais foram necessários e ao meu afilhado Caio Victor, pelas vezes que não me deixou estudar para brincar com ele, e assim conseguir me distrair e tornar essa fase mais leve. Amo vocês.

Às minhas avós, Mariita e Marieta (*in memorian*), que, mesmo ausentes fisicamente, sei que a conclusão desta etapa as deixarão orgulhosas de onde estiverem e à todos os meus tios, tias e primos que sempre torceram por mim.

Ao meu namorado Luís Eduardo, agradeço por acreditar e me fazer acreditar que posso mais do que imagino, por me cobrar sempre e pela paciência com minhas ausências e incentivos.

Às minhas grandes amizades, que se preocuparam, não me deixaram desistir ou simplesmente ouviram os meus desabafos. Muito obrigada Danny, Pablo, Jéssica, Bryanne, Andreza, Rayssa, Adriana e Luana.

Aos queridos Adicinetes: Italo, Otávio, Fernando e Diego pelo acompanhamento e por sempre se mostrarem dispostos a ajudar.

Aos tops de ES: Igor, Jorge, Fernanda e Adriana, pelas vezes em que compartilhamos nossas aflições e conquistas.

À Companhia Industrial Têxtil. Agradeço especialmente a Antônio João e Cláudio Júnior pelo apoio e compreensão para que eu pudesse realizar as atividades do mestrado.

Aos professores que tive desde a época da escola até a conclusão do mestrado e aos demais colegas de trabalho, graduação e mestrado. Enfim, a todos que me apoiaram, cobraram, auxiliaram e ajudaram nesse, que foi meu maior desafio.

RESUMO

O *Model Driven Development* (MDD) promove o uso de modelos na geração de soluções de *software*, na qual os modelos são o artefato principal do desenvolvimento. Nos últimos anos, estudos na área de Engenharia de *Software* (ES) tem sido intensificados e diversas soluções, definições, métodos e estruturas estão sendo geradas. Entretanto, esse paradigma, como é considerado por alguns autores, ainda não é utilizado de forma efetiva, visto que a ES abrange diversos aspectos e nem todos estão preparados para o MDD. A Engenharia de Requisitos (ER) ganha destaque nesse cenário, pois compreende a fase inicial do processo de desenvolvimento e o produto gerado deve atender aos requisitos definidos nesse momento. Uma vez que o código deixa de ser o foco do desenvolvimento no contexto de MDD, surge a necessidade de identificar se os requisitos definidos no início do projeto, ao serem transformados, mantém-se fieis, bem como identificar os elementos alocados a cada requisito em todas as fases e possíveis impactos em caso de alteração. Nesse contexto, torna-se importante introduzir os conceitos da rastreabilidade de requisitos em MDD. Dessa forma, este trabalho apresenta o R2MDD, um *framework* que visa monitorar e rastrear requisitos durante as transformações de modelos até a geração de código fonte. O R2MDD busca identificar impactos, garantir a consistência dos requisitos e demais características, bem como gerar informações a nível gerencial, que beneficiam todos os *stakeholders*. Um caso exemplo foi realizado com o auxílio do modelo *Qualitas* no Hospital Universitário da Universidade Federal de Sergipe (HU – UFS) a fim de avaliar o R2MDD, destacar suas vantagens e identificar suas limitações.

Palavras-chave: Desenvolvimento Dirigido a Modelos. Engenharia Dirigida a Modelos. Arquitetura Dirigida a Modelos. Engenharia de Requisitos. Rastreabilidade de Requisitos. Engenharia de *Software*.

ABSTRACT

The Model Driven Development (MDD) promotes the use of models to generate software solutions, in which models are the primary development artifact. In recent years, studies in software engineering area (ES) have been intensified and several solutions, definitions, methods and structures are being generated. However, this paradigm, as it is considered by some authors, has not been effectively used, as the ES covers various aspects, and not everyone is prepared for MDD. The Requirements Engineering (RE) is highlighted in this scenario, since it comprises the initial phase of the development process and the product generated must meet the requirements set out in that time. Once the code ceases to be the focus of development in the context of MDD, it is necessary to identify whether the requirements defined at the beginning of the project while under transformation remain faithful as well as the allocated elements to each requirement at all stages and possible impacts in case of change. In this context, it is important to introduce the concepts of traceability requirements in MDD. Thus, this work presents the R2MDD, a framework which aims to monitor and track requirements for the transformation of models to the generation of source code. The R2MDD seeks to identify impacts, ensure the consistency of requirements and other characteristics, as well as generate information management level, benefiting all stakeholders. An experimental study was conducted with the assistance of Qualitas model at the University Hospital of the Federal University of Sergipe (HU - UFS) to assess the R2MDD, highlighting its advantages and limitations.

Key-words: Model-Driven Development. Model-Driven Engineering. Model-Driven Architecture. Requirements Engineering. Requirements Traceability. Software Engineering.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE SIGLAS

CIM	*Computation Independent Model*
CITM	*Computation Tests Indepent Model*
ER	Engenharia de Requisitos
ES	Engenharia de *Software*
HU	Hospital Universitário
HU – UFS	Hospital Universitário da Universidade Federal de Sergipe
IEEE	*Institute of Electrical and Electronics Engineers*
MDA	*Model-Driven Architecture*
MDD	*Model-Driven Development*
MDE	*Model-Driven Engineering*
MMR	Metamodelo de Rastreabilidade
MOF	*Meta Object Facility*
M2M	*Model to Model*
M2T	*Model to Text*
NFR	Requisitos Não-Funcionais
OMG	*Object Management Group*
PIM	*Platform Independent Model*
PITM	*Platform Tests Independent Model*
PNTN	Programa de Triagem Neonatal
PSM	*Platform Specific Model*
PSTM	*Platform Tests Specific Model*
RN	Recém Nascido
RSL	Revisão Sistemática de Literatura
STRN	Serviço de Referência de Triagem Neonatal
SUS	Sistema Único de Saúde
UFS	Universidade Federal de Sergipe
UML	*Unified Modeling Language*

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

Este capítulo traz uma breve contextualização relacionada ao tema desta pesquisa na Seção 1.1. Em seguida, na Seção 1.2, são apresentadas a problemática e a motivação encontradas para realização deste trabalho. Na Seção 1.3, é levantada a hipótese que se pretende provar e na Seção 1.4 são discutidos os objetivos. As contribuições esperadas são descritas na Seção 1.5 e após isso, a Seção 1.6 apresenta a metodologia de pesquisa utilizada. Por fim, a seção 1.7 apresenta a organização da dissertação.

1.1 Contextualização

A crescente necessidade de informação que ocorre devido à alta competitividade exige que os sistemas de informações atuais sejam mais rápidos e confiáveis. Além disso, eles precisam satisfazer as expectativas de seus clientes e usuários, as quais se encontram cada vez mais ambiciosas. Segundo Pressman (2016), para atender a essas expectativas alguns desafios devem ser enfrentados, como: aumento da complexidade e comunicação entre aplicações, alterações frequentes dos requisitos, maior qualidade e menor custo, cumprimento de prazos e surgimento de novas tecnologias.

Devido a isso, a Engenharia de *Software* (ES) está passando por constantes transformações com o objetivo de atender às necessidades das pessoas e organizações. A evolução das abordagens dirigidas a modelos ganha destaque nesse cenário. A ideia central é utilizar modelos como artefato principal do desenvolvimento com o objetivo de gerar *softwares* sem o envolvimento dos conceitos específicos de plataformas (ZOHREVAND *et al.*, 2011).

Essas abordagens fornecem meios promissores para automatizar o processo de *software* e, por esse motivo, são mais flexíveis para lidar com os desafios. Segundo Chitforoush *et al.* (2007), esse fator promete reduzir os custos de desenvolvimento, bem como

melhorar a consistência do *software*, manutenção e qualidade. Elas podem ainda oferecer benefícios como: aumento da produtividade e portabilidade e facilidade na evolução e manutenção do *software*.

Uma dessas abordagens é o Desenvolvimento Dirigido a Modelos (MDD), que busca promover o uso intensivo de modelos no desenvolvimento de *software*. Segundo Valderas e Pelechano (2009), em MDD os modelos auxiliam na compreensão de problemas complexos e suas soluções através de abstrações. Para os autores, os processos em MDD tem início com a geração de um modelo de requisitos, que descreve as necessidades do usuário e o papel do sistema de uma forma independente de computação.

Esse modelo é refinado e transformado em um ou mais modelos conceituais sem considerar os aspectos tecnológicos, que são transformados em outros modelos que descrevem o sistema com base em uma plataforma específica para, por fim, ser gerado o código-fonte. O refinamento dos requisitos e modelos ocorre por meio de transformações de modelo para modelo e modelo para código. As transformações devem fornecer uma solução baseada em modelos de um problema considerado clássico da ES: como ir do espaço do problema (requisitos do usuário) ao espaço da solução (concepção e implementação)?

Ainda para Valderas e Pelechano (2009), essas transformações devem estabelecer de maneira clara como os requisitos definidos no início do processo são transformados em modelos conceituais e como esses modelos conceituais são transformados em código. Porém, para os autores, o MDD ainda apresenta algumas restrições quanto a esse fator.

Em geral, durante as transformações nenhuma informação é fornecida para indicar se os requisitos permanecem fidedignos do início ao fim do processo. Esse aspecto dificulta a validação das transformações e a verificação se elas foram realizadas da maneira correta e se os elementos derivados durante o processo de transformação atendem aos requisitos especificados.

Na visão de Seibel *et al.* (2010), esse é um problema que pode ser resolvido por meio da introdução dos conceitos de rastreabilidade e monitoramento dos requisitos em MDD. De acordo com os autores, a rastreabilidade de requisitos é uma etapa do processo de Gerência de Requisitos da Engenharia de Requisitos (ER) que se refere à capacidade de descrever o ciclo de vida do requisito.

Para Sayão e Leite (2005), a ER deve ser utilizada para fornecer os relacionamentos entre os requisitos, arquitetura e implementação final do sistema, bem como viabilizar um entendimento apropriado dos relacionamentos de dependência dos requisitos. Ainda segundo

os autores, é por meio desses relacionamentos que o projetista identifica se os requisitos estão em conformidade com o que foi especificado e se é possível perceber de maneira precoce se determinado requisito será atendido pelo *software*.

Segundo Sayão e Leite (2005), a rastreabilidade dos requisitos é identificada como um fator de qualidade, podendo também auxiliar gerentes e desenvolvedores em diversas situações, como verificar a alocação dos requisitos a componentes de *software*, resolver problemas de requisitos em conflitos, identificar requisitos que não foram previstos, corrigir os problemas encontrados, facilitar a fase de validação do processo da ER, analisar o impacto na evolução do sistema, prever custos e prazos, gerenciar os riscos, reutilizar componentes, entre outros.

Alinhada aos benefícios prometidos, a rastreabilidade de requisitos pode ser eficaz se aplicada aos processos do MDD. Isso porque ela irá auxiliar no controle e monitoramento dos requisitos durante as transformações servindo de apoio a diversos tipos de análise, como por exemplo, averiguar o estado dos requisitos durante as transformações de modelos, bem como verificar se os requisitos mantêm-se fiéis após cada transformação, além de determinar se um sistema corresponde ao que foi definido ou identificar se o mesmo está em conformidade.

1.2 Problemática e Motivação

Segundo Valderas e Pelechano (2009), em MDD as transformações entre os modelos estabelecem de forma clara como os requisitos são transformados em modelos conceituais e depois em código. Entretanto, ainda não é dado um *feedback* sobre como essas transformações são aplicadas e como os requisitos se comportam ao longo do processo. Esse aspecto gera um problema: a dificuldade em validar se os elementos derivados durante os processos satisfazem corretamente os requisitos. Uma opção para tratar esse problema seria a aplicação da rastreabilidade de requisitos para o MDD. Entretanto, de acordo com Gotel e Finkelstein (1994), diversos desafios e problemas são identificados quanto ao suporte à prática da rastreabilidade, tais como:

- Rastreabilidade sendo vista como uma atividade repetitiva e entediante;
- Distribuição da responsabilidade pela rastreabilidade a todos os membros do projeto;

- Identificar o que deve ser rastreado e como as relações devem ser estabelecidas;
- Relações de rastreabilidade tendem a enfraquecer se não houver manutenção constante; e,
- Descobrir como combinar abordagens humanas e automáticas heterogêneas que suportem a rastreabilidade.

Uma Revisão Sistemática de Literatura[1] (RSL) foi realizada com o objetivo de identificar pesquisas que tratem da rastreabilidade e monitoramento de requisitos para o MDD. Os resultados obtidos pela aplicação dos métodos de seleção dos trabalhos são mostrados de forma quantitativa na Tabela 1.1, sendo que a primeira coluna apresenta as etapas do protocolo, enquanto as demais apresentam as fontes de pesquisa utilizadas para a investigação e a quantidade de artigos encontrada em cada etapa e fonte.

Tabela 1.1: Resultado quantitativo dos artigos encontrados.

Etapas do Protocolo	**IEEE**	**ACM**	**SPRINGER**
Consulta das *strings* de busca	51	33	90
Período de 2005 a 2015	51	33	90
Critérios de inclusão	3	3	7
Leituras dos resumos	2	3	4
Critérios de Exclusão	2	3	2

Pode-se observar por meio da Tabela 1.1 que a quantidade de artigos encontrados com a aplicação das *strings* de busca é considerada baixa e que não foi encontrado nenhum artigo com período anterior ao ano de 2004.

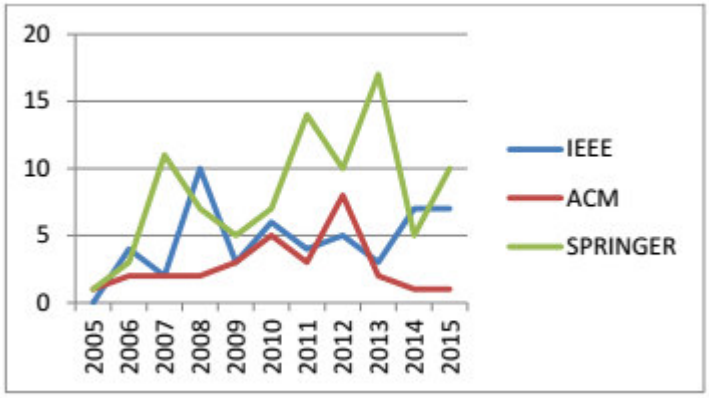
[1] Strings de busca utilizadas:
("model-driven approach" or "model driven approach") and ("traceability requirements" or "trace requirements");
("model-driven development" or "model driven development" or "mdd") and ("traceability requirements" or "trace requirements");
("model-driven architecture" or "model driven architecture" or "mda") and ("traceability requirements" or "trace requirements"); e,
("model-driven engineering" or "mde" or "model driven engineering") and ("traceability requirements" or "trace requirements").
Critérios de Inclusão:
Devem apresentar textos completos dos estudos em formato eletrônico;
Devem estar descritos em inglês; e,
Devem apresentar problemas da rastreabilidade de requisitos para as abordagens dirigidas a modelos.
Critérios de Exclusão: Trabalhos curtos e/ou de pouca relevância e/ou incompletos; Trabalhos repetidos; e, Trabalhos referentes a um mesmo estudo de fontes diferentes.

Isso se deve ao fato de que, apesar da rastreabilidade ser um assunto discutido há diversos anos, a preocupação com a rastreabilidade de requisitos com o foco em MDD é recente e tem ganhado destaque a partir de 2010.

Os trabalhos selecionados após a aplicação dos critérios de inclusão e exclusão são: ALMEIDA *et al.* (2006), VALDERAS e PELECHANO (2008), WINKLER e PILGRIM (2009), MERILINNA e YRJONEN (2010), LONIEWSKI *et al.* (2011), SANCHEZ *et al.* (2011) e SIEGERT *et al.* (2013).

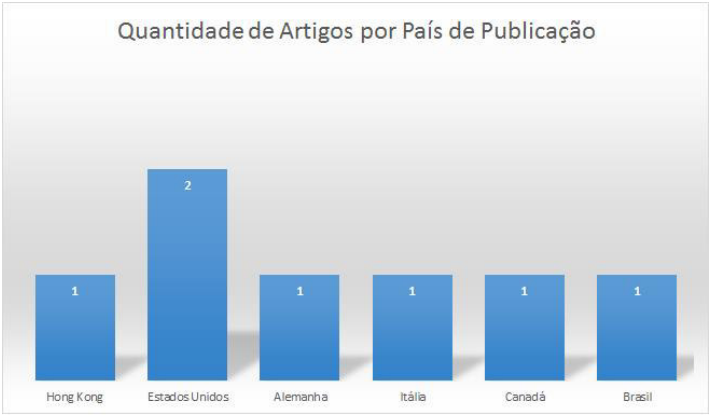
O Gráfico 1.1 apresenta a quantidade de trabalhos encontrados no período pesquisado por fonte de pesquisa. É possível inferir do gráfico que a distribuição da publicação de artigos sobre a rastreabilidade de requisitos em MDD tem variado bastante ao longo dos anos.

Gráfico 1.1: Resultado quantitativo de publicações por ano e fontes de pesquisa.



A quantidade de trabalhos selecionados por país de publicação está ilustrada no Gráfico 1.2, sendo três artigos publicados na América do Norte, dois na Europa, um na Ásia e um na América Latina.

Gráfico 1.2: Resultado quantitativo de publicações por local de pesquisa.



Almeida *et al.* (2006) propõem um *framework* com base na rastreabilidade de requisitos que tem como objetivo relacioná-los para os vários produtos do processo de MDD. Também é proposta uma noção de conformidade entre modelos que simplifica a rastreabilidade. Entretanto, o *framework* não se preocupa com a rastreabilidade de requisitos face às mudanças nas especificações dos mesmos.

Valderas e Pelechano (2008) apresentam uma abordagem que introduz as capacidades da rastreabilidade de requisitos no contexto de MDD. Os autores definem uma transformação modelo-para-modelo que fornece um *feedback* sobre como os modelos são aplicados. Essa transformação auxilia o analista a identificar se as demais transformações estão sendo realizadas de maneira correta. Além disso, os autores apresentam uma ferramenta de suporte estratégica para execução dos mapeamentos de maneira automática e que gera relatórios de rastreabilidade após a aplicação das transformações.

Winkler e Pilgrim (2009) fazem um levantamento do estado da arte e identificam os fatores limitantes da rastreabilidade de requisitos para o MDD. Para os autores, existem algumas restrições que dificultam a aplicação de práticas de rastreabilidade em MDD. Além disso, eles mencionam que as ferramentas de rastreabilidade atuais não oferecem suporte a MDD e consequentemente *links* de rastreabilidade podem não ser colocados em uso. O trabalho fez um levantamento das dificuldades da rastreabilidade de requisitos para o MDD, porém não apresentou nenhuma proposta para resolver esses problemas.

Merilinna e Yrjonen (2010) apresentam uma solução integrada de ferramentas para uma abordagem de modelagem específica de domínio que permite e orienta a definição de requisitos não-funcionais (NFR) precisos e não conflitantes. Ao adaptar os NFRs com o *Framework* e a integração com ferramentas de gerenciamento de requisitos, foi realizado um teste de rastreabilidade de NFR desde os requisitos iniciais até a concepção. A solução é baseada em uma abordagem de modelagem integrada, onde as técnicas de modelagem são aplicadas desde o início do processo.

O trabalho de Loniewski *et al.* (2011) apresenta uma abordagem metodológica ágil de ER dirigida a modelos e orientada à arquitetura que promete benefícios como a rastreabilidade mais consistente, melhor arquitetura e coerência da implementação. O trabalho preocupa-se com a rastreabilidade dos requisitos em MDD, esse, no entanto, não é o artefato principal e não há uma proposta focada em rastreabilidade.

Sanchez *et al.* (2011) propõem um ambiente integrado que envolve recursos que garantem a implementação de sistemas que apoiam a MDE, segurança e rastreamento de

requisitos com foco em sistemas industriais robóticos. Além disso, os autores propõem um *framework* para a gestão de requisitos de segurança e uma abordagem de rastreabilidade em um ambiente MDE. O trabalho contribui ao fornecer um metamodelo que considere o aspecto da segurança no desenvolvimento de aplicações para robôs; fornecer um metamodelo e um conjunto de transformações do modelo para derivar automaticamente implementações orientadas a objetos a partir de descrições arquiteturais baseadas em componentes; e, fornecer uma ferramenta que gere um relatório de rastreabilidade de transformações de modelos.

Siegert *et al.* (2013) apresentam uma abordagem orientada a modelos denominada MDEReq que possui foco na especificação e na gerência de requisitos no âmbito dos *softwares* embarcados. Apresentam ainda uma ferramenta de suporte a essa abordagem que realiza automatização da gestão de requisitos, bem como a gestão automática das matrizes de rastreabilidade de requisitos. Essa ferramenta foi denominada MDEReqTraceTool. Para provar a utilidade da ferramenta os autores também aplicaram um estudo de caso. Apesar da proposta de uma ferramenta, a mesma limita-se à abordagem MDEReq.

Esses trabalhos fazem diversas propostas para associar a rastreabilidade de requisitos no contexto de MDD. Entretanto, eles possuem fatores limitantes e, assim, restam alguns questionamentos:

- Como identificar se os requisitos definidos inicialmente serão rastreados quando os artefatos principais são os modelos?
- Como garantir a integridade desses requisitos?
- Como identificar quais são os recursos alocados aos requisitos e como eles se relacionam?
- Como identificar quais recursos alocados aos requisitos serão afetados quando forem necessárias mudanças em seu contexto?

Considerando os fatores que motivaram este trabalho, destacam-se as dificuldades associadas à rastreabilidade e monitoramento de requisitos em MDD, que são: a necessidade de compressão básica em relação à variabilidade de requisitos durante as transformações de modelos e o pouco apoio para o gerenciamento de requisitos em MDD.

Sendo assim, a motivação desta dissertação é, por meio da proposição de um *framework*, contribuir para a melhoria da qualidade e produtividade nos processos de MDD, buscando meios de solucionar este problema por meio da aplicação dos conceitos de rastreabilidade e monitoramento de requisitos.

1.3 Hipótese

Este trabalho trata da hipótese de que a integridade dos requisitos, especificados nos modelos gerados, pode ser alcançada com o monitoramento e a rastreabilidade destes em todas as etapas do processo de desenvolvimento.

1.4 Objetivos

Esta seção descreve os objetivos desta dissertação, destacando o objetivo geral e também os específicos.

1.4.1 Objetivo Geral

O objetivo deste trabalho é contribuir para a Engenharia de *Software* quando o foco é o MDD, por meio da proposta do *framework* R2MDD, que visa apoiar a identificação e monitoramento dos requisitos em todas as etapas do processo de desenvolvimento de *software*.

1.4.2 Objetivos Específicos

Para alcançar o objetivo proposto, alguns objetivos específicos foram definidos, são eles:

- Executar uma revisão sistemática de literatura que buscou identificar propostas que tratassem da rastreabilidade de requisitos para o MDD a fim de encontrar o estado da arte;
- Mapear os problemas relacionados à rastreabilidade de requisitos para o MDD, com o intuito de analisar as lacunas existentes;
- Identificar os principais pontos de convergência entre o MDD e a rastreabilidade de requisitos tendo em vista estabelecer o alinhamento entre esses conceitos;

- Definir o R2MDD com o propósito de apoiar a rastreabilidade e o monitoramento dos requisitos durante as etapas do MDD; e,
- Realizar um caso exemplo com o *framework* proposto com o intuito de validação.

1.5 Contribuições Esperadas

O R2MDD busca garantir o possível controle dos requisitos desde o início do projeto até a geração do código fonte. Espera-se que, desta forma, ele contribua com a melhoria da qualidade e com o aumento da produtividade ao reduzir o tempo de verificação e validação do *software* e melhorar a confiabilidade dos requisitos. Assim também traz uma discussão quanto à importância da aplicação dos conceitos da ER quando adotadas as abordagens dirigidas a modelos, visto que esse é um cenário que promete estar em predominância nos próximos anos.

Além disso, a ES pode ser beneficiada com a utilização do *framework*, uma vez que o R2MDD busca facilitar e gerar benefícios de apoio ao MDD em longo prazo. Esses benefícios são herdados das situações que a rastreabilidade facilita que podem ser:

- Verificação da alocação dos requisitos a componentes do *software*;
- Resolução de conflitos entre os requisitos;
- Verificação dos requisitos no momento dos testes;
- Correção dos problemas encontrados por meio da identificação do componente que originou o erro;
- Validação do sistema junto ao cliente;
- Análise de impacto;
- Previsão de custos e prazos; e,
- Gerenciamento de riscos.

Com isso, gerentes de projetos e outras partes interessadas também podem ser favorecidos, como os analistas, *designers*, mantenedores, testadores e usuários finais.

1.6 Metodologia de Pesquisa

O enquadramento metodológico deste trabalho está alinhado às seguintes modalidades de pesquisa científica:

- Quanto à natureza de pesquisa: Pesquisa aplicada, que sugere a aplicação prática ligada à solução de problemas concretos (RODRIGUES, 2007);
- Quanto à abordagem do problema: Pesquisa qualitativa, em que dados obtidos são analisados de maneira indutiva (RODRIGUES, 2007);
- Quanto aos objetivos: Pesquisa exploratória, que se refere à caracterização, classificação e definição do problema;
- Quanto aos procedimentos técnicos: Caso exemplo, que visa provar na prática o que prega a teoria de forma sistemática, computável e controlada para avaliação da atividade humana.

Esse trabalho foi realizado de acordo com o seguinte roteiro:

- Revisão Bibliográfica por meio de uma pesquisa em materiais bibliográficos constituídos de artigos, livros, dissertações e teses;
- Estudo e análise de propostas que envolvem a ER e MDD com foco em rastreabilidade de requisitos por meio de uma revisão sistemática;
- Definição de uma *framework* de rastreabilidade de requisitos com foco em MDD; e,
- Análise e limitações e a validação da proposta.

1.7 Organização da Dissertação

O texto dessa dissertação está organizado em 5 capítulos. Os tópicos a seguir descrevem o conteúdo de cada um deles:

- Capítulo 1: Corresponde a esta introdução. Trata da contextualização, problemática e motivação, hipótese, objetivos, contribuições esperadas, metodologia de pesquisa e organização desta dissertação;
- Capítulo 2: Neste capítulo são discutidos os conceitos relacionados às Abordagens Dirigidas a Modelos, descrevendo a evolução e as principais

abordagens, bem como uma visão a respeito da visibilidade dos requisitos em MDD;

- Capítulo 3: O capítulo apresenta uma revisão dos conceitos e fundamentos referentes a ER, abordando suas fases com enfoque na rastreabilidade. Também é feita uma análise das propostas para rastreabilidade com foco em MDD;
- Capítulo 4: Descreve o R2MDD, sua estrutura, etapas e atividades;
- Capítulo 5: É apresentada a validação e simulação do R2MDD para avaliação da sua viabilidade através de um caso exemplo; e,
- Capítulo 6: Relata as considerações finais, principais contribuições, limitações do trabalho e trabalhos futuros.

CAPÍTULO 2

ABORDAGENS DIRIGIDAS A MODELOS

A ES tem passado por grandes transformações ao longo do tempo. Os clientes e usuários estão ainda mais exigentes, bem como há a grande necessidade de se reduzir o tempo de desenvolvimento. Além disso, o processo de desenvolvimento de *software* tem se tornado ainda mais complexo, envolvendo o conhecimento e tecnologias específicas como linguagens de programação, ambientes de desenvolvimento e paradigmas de desenvolvimento, entre outros (OLIVEIRA, 2012).

Nesse contexto, surgem propostas que mudam o foco da programação a partir da aplicação de diferentes níveis de modelagem e modelos de transformação representados pelas Abordagens Dirigidas a Modelos (ZOHREVAND *et al*., 2011). Essas abordagens prometem benefícios como: custo reduzido de desenvolvimento, melhor qualidade do produto, maior retorno em relação aos investimentos em tecnologia, entre outros (ALMEIDA, 2014).

Esse capítulo discorre sobre os principais conceitos das Abordagens Dirigidas a Modelos, entre eles: *Model Driven Engineering* (MDE), *Model Driven Development* (MDD) e *Model Driven Architecture* (MDA).

2.1 Conceitos Básicos sobre as Abordagens Dirigidas a Modelos

As técnicas e ferramentas de desenvolvimento de *software* encontram-se em constante desenvolvimento. Apesar da evolução, ainda existe uma grande preocupação com modelagem, manutenção, documentação, validação, reuso, entre outros fatores necessários em projetos de *software* (OLIVEIRA, 2012). Devido a isso, surgem as abordagens dirigidas a modelos com a finalidade de melhorar a produtividade sem a redução da qualidade do *software*.

Um dos princípios da ES é a abstração (PRESSMAN, 2016). Kramer e Hazzan (2006) afirmam que a abstração é um meio cognitivo que possui a finalidade de superar a complexidade em um estágio específico da solução de um problema, concentrando-se em

características especiais e ignorando detalhes irrelevantes. Para os autores, o objetivo é simplificar a análise separando os atributos que são importantes dos que não o são. Sob o ponto de vista de Gharaat *et al.* (2015), o termo abstração representa simplificação. Para os autores, essa simplificação pode ser feita por meio de duas tarefas importantes: redução dos detalhes e generalização, que é a identificação e especificação das características comuns e importantes.

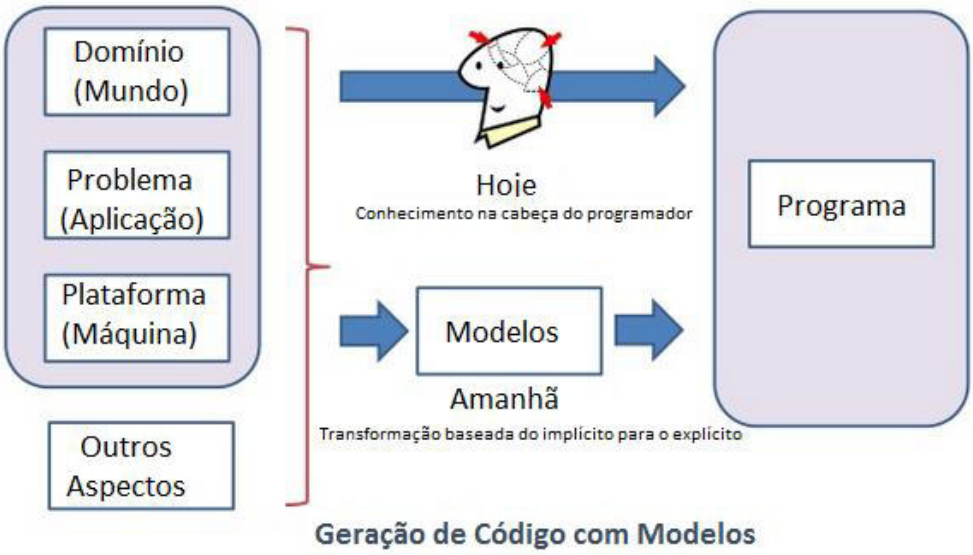
A abstração é utilizada para entender e construir modelos. Segundo Swithinbank (2005), os modelos podem ser considerados a representação de um sistema a partir de uma perspectiva própria, com a omissão de detalhes com pouca relevância, de modo que as características sejam percebidas com clareza. Já na visão de Almeida (2014), o modelo é um conceito simples do mundo real, que foi estabelecido para proporcionar um melhor entendimento de um *software* a ser desenvolvido.

Os modelos são criados por meio de uma linguagem de modelagem que oferece uma sintaxe e semântica que regulamentam os elementos e suas relações. Em desenvolvimento de *software,* os modelos construídos podem ser representados pela linguagem *Unified Modeling Language* (UML) (SWITHINBANK, 2005).

No âmbito de Abordagens Dirigidas a Modelos, os modelos não são utilizados somente como esboço de um projeto, mas como componente principal a partir do qual as implementações são geradas (SWITHINBANK *et al.,* 2005). Para Maciel (2010), as abordagens dirigidas a modelos caracterizam-se pela utilização de modelos que representam as informações de um sistema e orientam em sua implementação, oferecendo vantagens como geração automática do código a partir de modelos. Já na visão de Ameller (2009), as tecnologias de engenharia orientadas a modelos proporcionam uma abordagem propícia a lidar com a inaptidão das linguagens de terceira geração para amenizar a complexidade das plataformas existentes e os conceitos de domínio.

Bézivin (2014) descreve (Figura 2.1) que o conhecimento do domínio, problema e plataforma no cenário atual está sob a responsabilidade do programador, e, apresenta uma perspectiva desse cenário com a utilização do MDD. O autor considera que atualmente todas as informações relacionadas ao domínio, problema, plataforma e outros aspectos do *software* estão armazenadas na mente programador. Pode-se inferir da figura que com a utilização de modelos, a retenção de conhecimento do programador diminuirá, uma vez que as transformações, manutenções e inclusões de novas regras ocorrerão por meio de modelos.

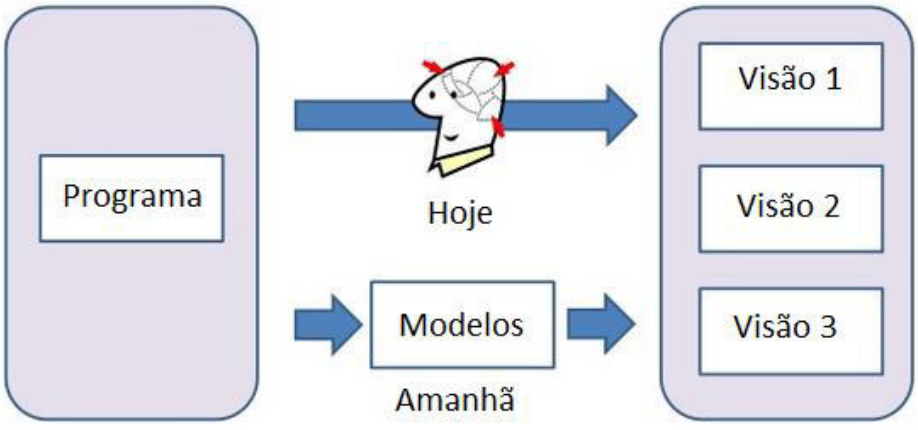
Figura 2.1: Modelos para produzir programas.



Fonte: Traduzido de Bézivin (2014).

Bézivin (2014) também apresentou (Figura 2.2) como os demais usuários e os *stakeholders* conseguem compreender os objetivos do *software* em ambos os cenários, sob o contexto atual e com a utilização de modelos. No primeiro caso as visões existentes são baseadas no conhecimento do programador, já com a utilização de modelos ocorrerá uma independência do programador uma vez que o conhecimento ficará por conta de modelos que possuem as regras de negócio e são auto explicativos.

Figura 2.2: Modelos para compreensão de programas.



Fonte: Traduzido de Bézivin (2014).

Na visão de Tankovic (2014), apenas a utilização de modelos em desenvolvimento de *software* não garante o sucesso do projeto. O autor afirma que para uma abordagem

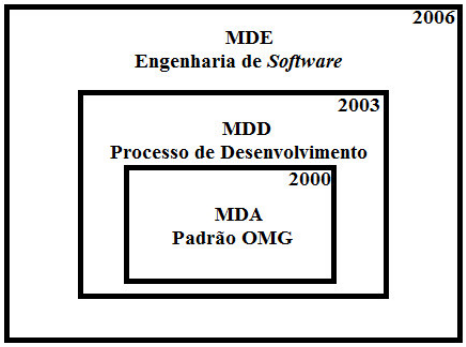
orientada a modelos ser eficaz e útil existem algumas características que devem levadas em consideração, sendo:

- Abstração: redução da complexidade com a omissão de detalhes irrelevantes segundo um ponto de vista;
- Compreensibilidade: representação clara de um modelo por meio de um formulário cuidadosamente escolhido com o objetivo de reduzir o esforço para o entendimento; e,
- Custo reduzido: rara alcançar os objetivos é exigido um menor espaço de tempo e redução de recursos financeiros.

Ainda segundo Tankovic (2014), deve-se escolher um nível adequado de abstração para a obtenção da simplicidade, uma vez que o excesso de abstração pode produzir soluções com flexibilidade inadequada enquanto que modelos menos abstratos podem ser de difícil compreensão.

A área da Engenharia de *Software* que trata o modelo como artefato principal do projeto de *software* é a *Model Driven Engineering* (MDE) (DEMIR *et al.,* 2006). Woodside *et al.*(2014) afirmam que a MDE faz uso da abstração com o objetivo de separar o modelo do software a partir de modelos de plataforma subjacentes e também realiza a automação para gerar código a partir de modelos. Além disso, para os autores, a MDE busca facilitar a análise das propriedades não-funcionais, tais como desempenho, escalabilidade, fiabilidade, segurança, entre outros. As técnicas de MDE tornaram-se úteis não somente para o desenvolvimento de novos *softwares*, como para a reengenharia de sistemas legados. Para os autores, as ideias principais do MDE são: elevar o nível de abstração e o grau de automação do computador. A classificação evolutiva das abordagens é apresentada por Ameller (2009) (Figura 2.3).

Figura 2.3: Iniciativas dirigidas a modelos.



Fonte: Traduzido de Ameller (2009).

O *Model Driven Development* (MDD) faz parte do contexto da MDE, já que ele é considerado um paradigma de desenvolvimento de *software* onde os modelos desempenham o papel fundamental. Em MDD, os modelos são utilizados para especificar, simular, verificar, testar e gerar o sistema a ser construído (Ameller *et al.,* 2010).

A *Model Driven Architecture* (MDA), por sua vez, é um padrão arquitetural apresentado pelo *Object Management Group* (OMG) que propõe a definição de um conjunto de normas e propriedades que visam apoiar a construção do *software* por meio de uma arquitetura. Segundo o OMG (2014), a MDA trata desde os requisitos para a modelagem de negócios até as implementações de tecnologia.

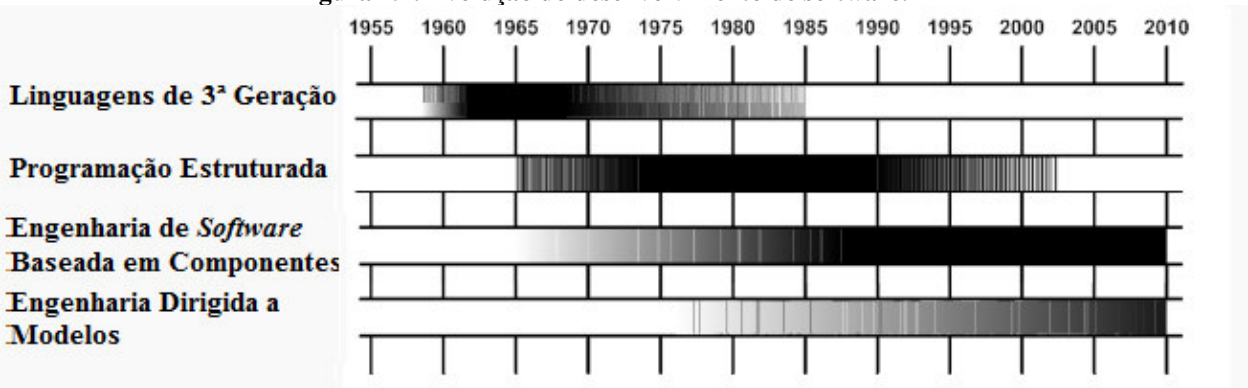
2.2 Evolução das Abordagens Dirigidas a Modelos

Para Pressman (2016), os últimos 50 anos foram marcados por modificações expressivas em relação ao papel evolutivo do *software*. Segundo Atkinson e Kuhne (2003), com o surgimento da primeira versão do compilador Fortran, pela primeira vez, foi permitido que os programadores especificassem o que a máquina deveria fazer ao invés de como ela deveria fazer. Desde então, os engenheiros e programadores continuaram trabalhando em ritmo acelerado para aumentar o nível de abstração da programação.

As abordagens dirigidas a modelos podem ser consideradas como uma continuação dessa tendência. Com elas, ao invés de exigir que os programadores comuniquem-se com o computador através de uma linguagem complexa, são utilizados modelos para especificar as funcionalidades de um sistema e gerar seu código-fonte (ATKINSON E KUHNE, 2003).

Por encontrar-se sempre em constante mudança (Figura 2.4), cada novo avanço no desenvolvimento de *software* caracteriza-se como uma melhoria no nível de abstração que tende à independência de plataforma e portabilidade do *software* (AMELLER, 2009).

Figura 2.4: Evolução do desenvolvimento de software.

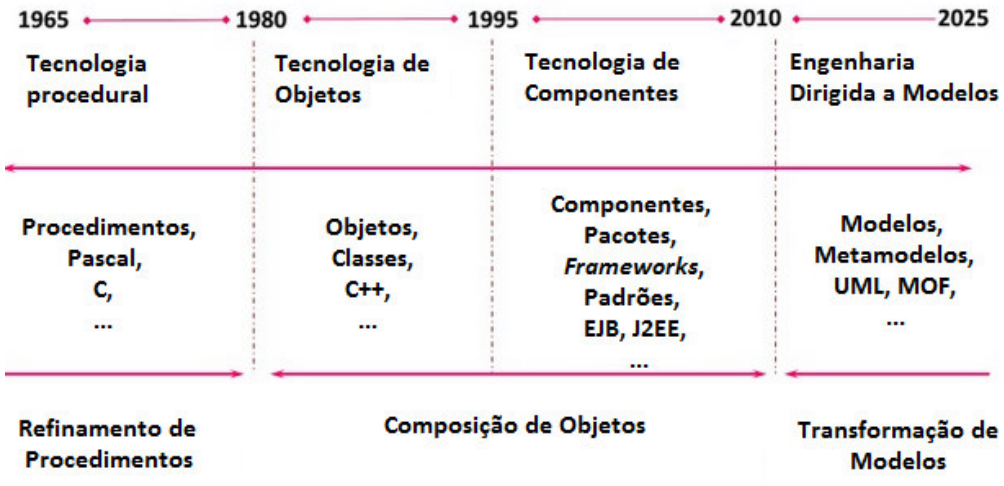


Fonte: Traduzido de Ameller (2009).

As partes mais escuras da Figura 2.4 representam um período com maior intensidade da utilização de cada abordagem citada. Pode-se observar na Figura que a MDE começa a ganhar destaque a partir do ano 2000, mas ainda não é um conceito muito difundido, prevalecendo a Engenharia Baseada em Componentes.

Bezivin (2014) também apresenta uma visão histórica referente às mudanças de paradigmas desde 1965 e uma previsão de como a ES estará até 2025 (Figura 2.5). Para o autor, há uma evolução que se inicia com a tecnologia procedural e com o passar dos anos surgem as tecnologias de objetos e de componentes. É somente a partir de 2010 que o autor apresenta a MDE com predominância nos anos seguintes.

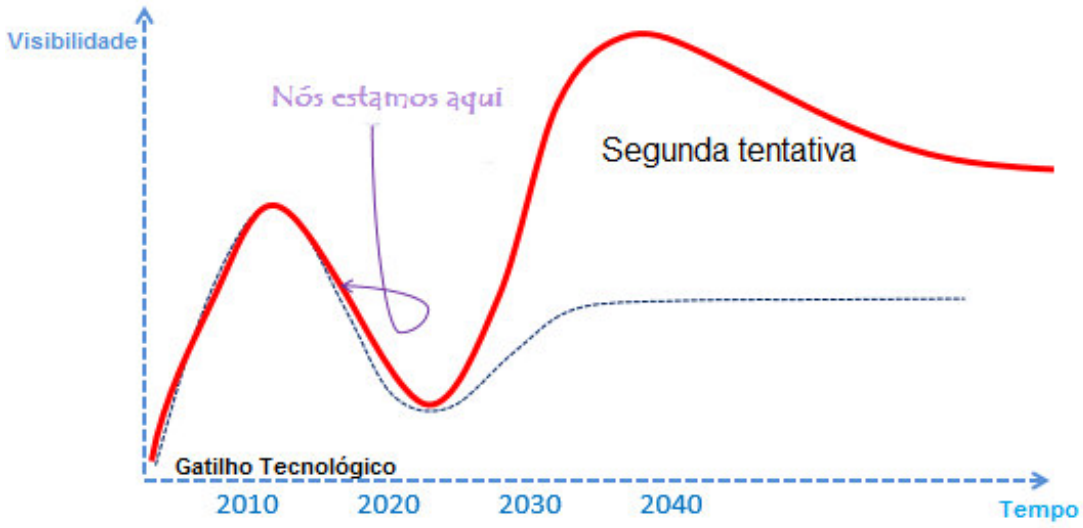
Figura 2.5: Mudanças de paradigmas/artefatos.



Fonte: Traduzido de Bézivin (2014).

Outra forma de observar a previsão da visibilidade do MDE nos próximos anos é apresentada por Bézivin (2014) através do gráfico da Figura 2.6.

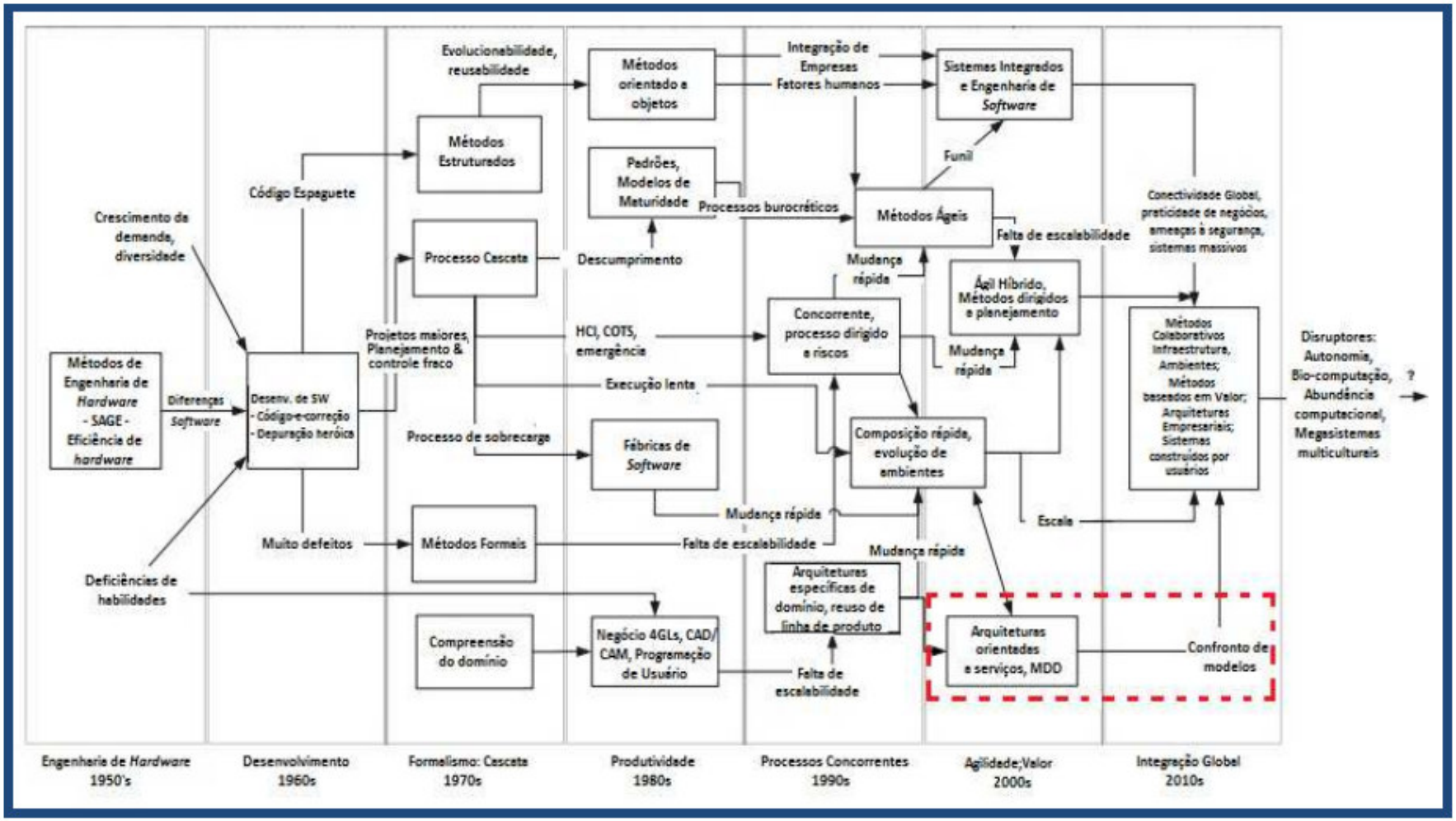
Figura 2.6: Possível cenário para a MDE.



Fonte: Traduzido de Bézivin (2014).

No contexto de MDD, Boehm (2006) apresentou (Figura 2.7) uma retrospectiva da evolução da ES nas últimas décadas e destacou as tendências para décadas futuras. Na Figura 2.7, pode-se observar que o MDD começou a ganhar força a partir de 2000 com a necessidade de maior agilidade e preocupação com os custos do *software*. Conforme Selic (2003), o termo MDD surgiu anos antes, mas como as técnicas de modelagem ainda eram restritas, ele obteve pouco sucesso inicial. Foi somente com o avanço tecnológico que o MDD passou a conquistar seu espaço.

Figura 2.7: Evolução e tendências da Engenharia de *Software*.



Fonte: Boehm (2006) *apud* Almeida (2014).

Cada dia que passa as abordagens dirigidas a modelos ganham mais relevância, porém ainda existem dificuldades e problemas a serem resolvidos, bem como ainda são poucas as iniciativas de adaptação para apoiar essas abordagens. Para Selic (2003), apesar da criação e utilização de novos paradigmas ao longo do tempo, o MDD representa a primeira grande mudança da geração de tecnologia de informação básica desde o surgimento dos compiladores. Ameller (2009), Boehm (2006) e Bézivin (2014) compartilham da mesma opinião. Para os autores, essa nova abordagem deverá ficar marcada como uma grande etapa da Engenharia de *Software*.

2.3 Desenvolvimento Dirigido a Modelos (MDD)

O MDD pode ser considerado um paradigma de desenvolvimento em que os modelos e suas transformações são o artefato principal do desenvolvimento de *software*. Sua característica principal é que o foco dos produtos primários de desenvolvimento são os modelos, ao invés de programas de computador (FRANCH *et al.*, 2010). No ponto de vista de Boehm (2006), o MDD possui uma perspectiva de desenvolvimento de modelos de domínio, cuja estrutura leva a arquiteturas de alta coesão e baixa modularidade, permitindo que a aplicação desenvolvida seja rápida e confiável.

Na visão de Lazarte (2010), o MDD utiliza os modelos de construção de *software* para simular, estimar, compreender, comunicar e produzir código. Já Franch *et al.* (2013) afirmam que os modelos são usados para especificar, simular, verificar, testar e gerar o sistema a ser construído. Selic (2003) menciona que uma das grandes vantagens do MDD é que ele oferece perspectivas de melhorias de compatibilidade, uma vez que sua proposta é que os modelos sejam independentes de plataforma e linguagens de programação.

Já segundo Atkinson e Kuhne (2015), o que motivou e impulsionou o MDD foi a proposta de melhor produtividade. Com isso, segundo os autores, os seguintes benefícios poderão ser percebidos:

- Melhoria em curto prazo da produtividade dos programadores, gerando o aumento do valor dos artefatos de *software* primários; e,
- Melhoria em longo prazo da produtividade dos programadores, reduzindo a velocidade com que os artefatos de *software* tornam-se obsoletos.

Segundo Swithinbank *el al.* (2005), a IBM também listou alguns benefícios que podem ser observados com a utilização do MDD. Entre os citados estão:

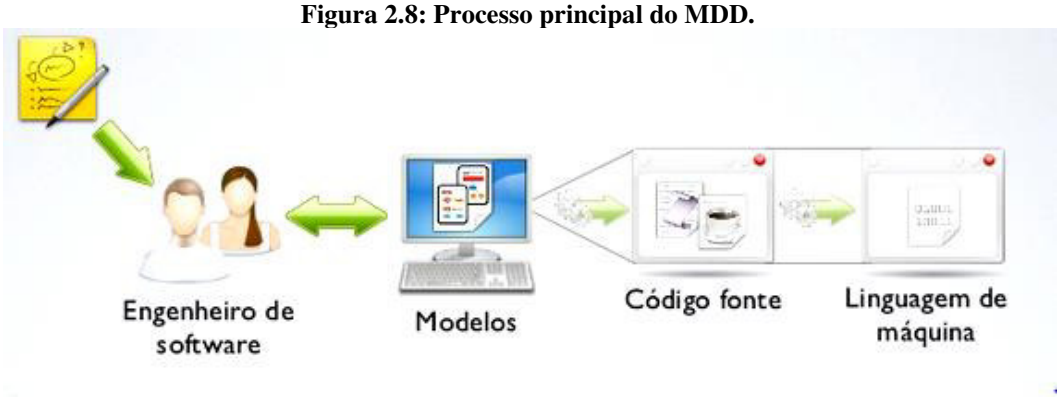
- Aumento da produtividade: MDD reduz o custo de desenvolvimento de *software* através de código e artefatos de modelos, o que aumenta a produtividade do desenvolvedor. É importante levar em consideração o custo do desenvolvimento, porém com um planejamento bem realizado, esse custo possivelmente será minimizado.
- Melhor manutenção: o MDD leva a uma arquitetura sustentável, em que as mudanças são feitas de forma rápida e consistente. Esse fator permite a migração mais eficiente dos componentes para novas tecnologias. Isso ocorre porque os modelos de alto nível são mantidos livres de detalhes de tecnologia, além disso, uma mudança de regra é realizada no modelo original e aplicada por meio de transformações.
- Reutilização de legados: é possível modelar sistemas legados em UML. Se existirem muitos componentes implementados no legado, podem-se realizar transformações inversas dos componentes para a UML. Nesse caso, tanto pode-se migrar os componentes para uma nova plataforma como o componente ser acessado via integração de tecnologias, por exemplo, os *Web Services*.
- Adaptabilidade: ao adicionar uma nova função ou regra, somente é necessário demonstrar o comportamento dessa funcionalidade, as demais informações necessárias para gerar os artefatos são obtidas durante as transformações.
- Aumento da comunicação com os *stakeholders*: como os modelos omitem os detalhes de implementação, os conceitos são facilmente compreendidos pelas partes interessadas. Esse fator facilita a comunicação com os *stakeholders* e a entrega de soluções alinhadas aos seus objetivos.
- Modelos como ativos de longo prazo: os modelos são ativos de Tecnologia de Informação, portanto, eles só mudam quando as regras de negócio mudam.
- Capacidade de adiar decisões de tecnologia: a escolha da plataforma e das tecnologias a serem utilizadas podem ser adiadas para quando houver mais informações disponíveis, uma vez que o foco inicial do MDD é a construção de modelos, então essas decisões não precisam ser tomadas em um momento inicial.

Para Selic (2003), a automação é considerada um meio eficaz de aumentar a produtividade e confiabilidade, mas o MDD somente alcançará os benefícios prometidos quando o seu potencial de automação for explorado de forma completa. As propostas

anteriores ao MDD limitavam-se à geração de modelos de apoio à geração de código, então a automação era pouco explorada.

Grillo (2012) aponta que modelos são a parte central do processo de desenvolvimento e é através deles que o código-fonte é gerado (Figura 2.8). Nesse caso, quando existe a necessidade de realizar alguma alteração de regra de negócio, esse ajuste deve ser feito no modelo, e não mais no código (GRILLO, 2012).

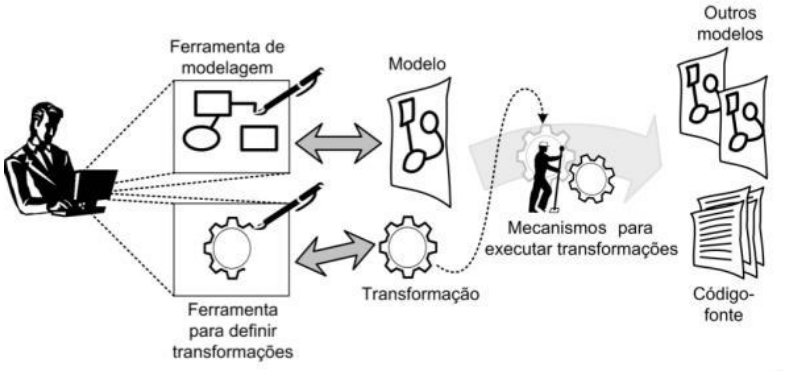
Figura 2.8: Processo principal do MDD.



Fonte: (GRILLO, 2012).

Por meio da Figura 2.9, Lucrédio (2009) apresenta os principais elementos utilizados em MDD.

Figura 2.9: Principais elementos de MDD.



Fonte: (LUCRÉDIO, 2009).

Lucrédio (2009) apresenta como principais elementos: ferramenta de modelagem, ferramenta para definir transformações, modelo, transformação, mecanismos para executar transformações, outros modelos e código-fonte. Através da ferramenta de modelagem o engenheiro irá construir os modelos que apresentam as regras e os conceitos do *software* a ser desenvolvido. Esses modelos servem como entrada para as transformações que irão gerar outros modelos ou o código-fonte. Essas transformações precisam ser definidas por uma ferramenta à qual o engenheiro possa informar as regras de transformação necessárias para a solução. Com isso é necessário um mecanismo que aceite o modelo como entrada e aplique as

transformações conforme as regras estabelecidas pelo engenheiro. Nesse momento deve existir uma documentação para auxiliar os desenvolvedores com a rastreabilidade e uma possível manutenção do *software* (LUCRÉDIO, 2009).

2.4 Arquitetura Dirigida a Modelos (MDA)

Mesmo com as ferramentas existentes, para que o MDD alcance seus objetivos é necessário lidar com alguns desafios, como as constantes mudanças na infraestrutura do *software* e reestruturações de domínio. Para isso, o *Object Management Group* (OMG), oferece uma estrutura conceitual para a definição de um conjunto de normas e padrões de apoio ao MDD: a MDA, que tem como ponto chave a UML e outras tecnologias relacionadas à modelagem. Além disso, a MDA pode utilizar outros padrões *Web* que também facilitam o uso do MDD, como o XML (SELIC, 2003).

Segundo a OMG (2003), a MDA provê uma abordagem que deriva valores a partir de modelos e uma arquitetura de apoio a todo o ciclo de vida das transformações. De acordo com o grupo, a MDA tem amplo suporte, desde a fase inicial de requisitos até a geração da tecnologia. Uma das propostas é que deve existir um entendimento comum dos padrões MDA pela comunidade e equipe de desenvolvimento. Para isso, a OMG (2003) propôs três formas de manter esse entendimento:

- Prover uma boa definição dos termos, ícones e notações para o entendimento comum de uma área específica;
- Fornecer uma base de modelos com dados semânticos a serem gerenciados em uma versão compartilhada; e,
- Prover bibliotecas de modelos reutilizáveis, com vocabulário e regras, processos reutilizáveis, modelos de negócios e padrões de projetos de arquitetura de comum entendimento.

Na visão de Franch *et al.* (2010), a MDA incentiva o uso de modelos e suas transformações em diferentes níveis. Os modelos abordados pela MDA são:

- *Computation Independent Model (CIM)*: visão de um sistema de um ponto de vista independente de computação. O CIM pode ser considerado o modelo de um negócio (AMELLER, 2010);

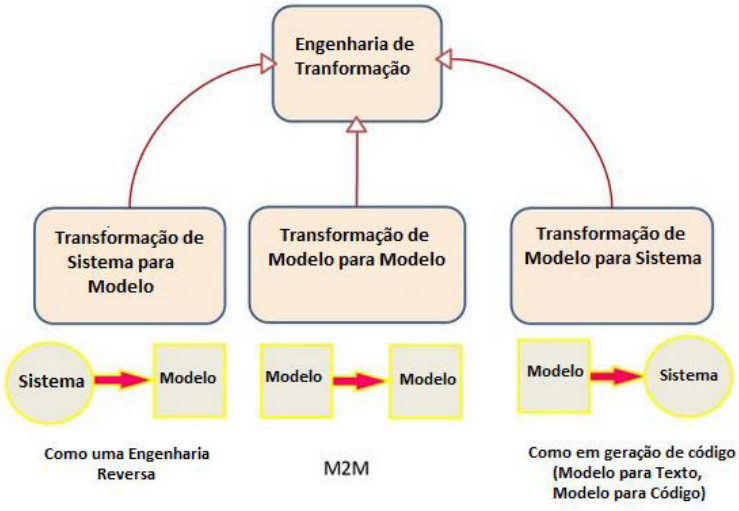
- *Platform Independent Model (PIM)*: especifica o sistema de *software* de uma maneira independente da plataforma da tecnologia escolhida para implementá-lo (FRANCH *et al.*, 2010); e,
- *Platform Specific Model (PSM)*: refina a PIM com a especificação da plataforma de desenvolvimento do *software*. Sendo assim, duas implementações diferentes do mesmo sistema compartilham o mesmo PIM, mas têm dois PSMs diferentes, cada uma adaptada para as capacidades tecnológicas de cada plataforma (FRANCH *et al.*, 2010).

As transformações mencionadas por Franch *et al.* (2010) são:

- *Model to Model (M2M)*: momento em que as transformações evoluem um PIM em um PSM; e,
- *Model to Text (M2T)*: utilizados para gerar o sistema executável a partir do PSM. Essa etapa inclui a geração de vários artefatos de código, como por exemplo, classes de negócio *Java*, esquemas de banco de dados *Oracle*, entre outros.

Bezivin (2014) apresenta as transformações possíveis em um contexto MDE (Figura 2.10). Seu objetivo é apontar o que o autor considera Engenharia de Transformação. Observa-se que a MDE proporciona tanto a transformação entre modelos, como a transformação de sistemas para modelos ou de modelos para sistema. Essa primeira pode ser considerada uma Engenharia Reversa, quando o objetivo é migrar *softwares* legados a fim de gerar um novo *software* por meio do MDD.

Figura 2.10: Transformações em um contexto MDE.



Fonte: Traduzido de Bézivin (2014).

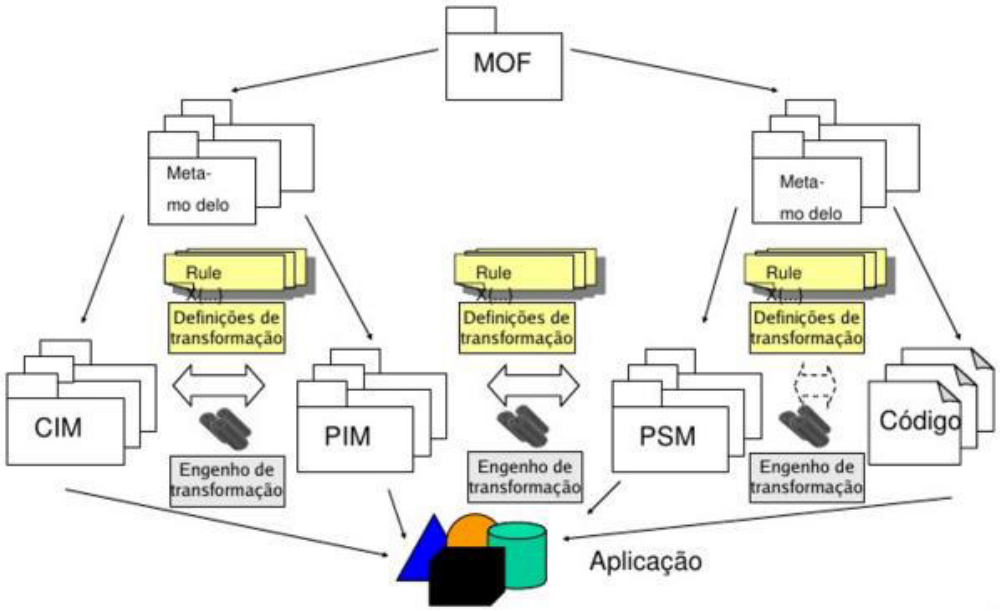
Ameller (2010) apresenta a atual metodologia de MDA com foco nessas transformações (Figura 2.11).

Figura 2.11: Fluxo de transformações comuns em MDA.

Fonte: (AMELLER, 2010).

Os retângulos presentes na Figura 2.11 representam as transformações M2M e M2T, enquanto que os círculos representam os modelos e, por fim, o código. As transformações entre modelos M2M são realizadas por meio de linguagens de transformação, enquanto que as transformações de modelo para texto, M2T, são realizadas por meio de linguagens de *script* (AMELLER, 2010). Segundo Vieira (2010), a geração do PIM tem como entrada o CIM, enquanto que a geração PSM tem como entrada o PIM. Por fim, o PSM serve como a entrada para a geração do código. Outra forma de visualizar essas transformações pode ser representada na Figura 2.12.

Figura 2.12: Arquitetura MDA.

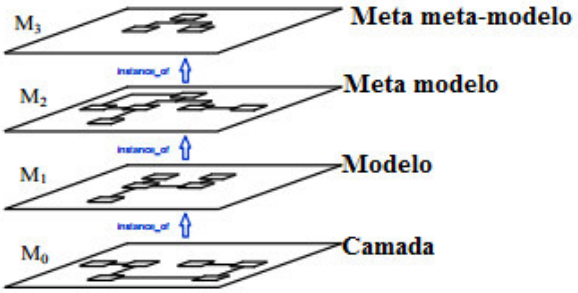


Fonte: (RAMALHO, 2010).

De acordo com Ramalho (2010), "os metamodelos são modelos que descrevem modelos". A UML é um exemplo de um metamodelo, uma vez que nela existem elementos que serão utilizados para criar os modelos das aplicações. Na visão da autora, as classes, atributos, associações e outros elementos podem ser utilizados porque são pré-definidos nesse metamodelo.

Diante disso, Sharma e Sood (2011) afirmam que os modelos precisam estar em concordância com seus respectivos metamodelos, que estão em conformidade com os meta-metamodelos. O *Meta Object-Facility* (MOF) é o mecanismo de definição da linguagem do MDA. Ele determina uma sintaxe abstrata de construções de modelagem para a especificação, construção e gestão de metamodelos tecnologicamente neutros. A Figura 2.13 apresenta uma MOF de quatro camadas de arquitetura.

Figura 2.13: Infraestrutura de modelagem tradicional da OMG.



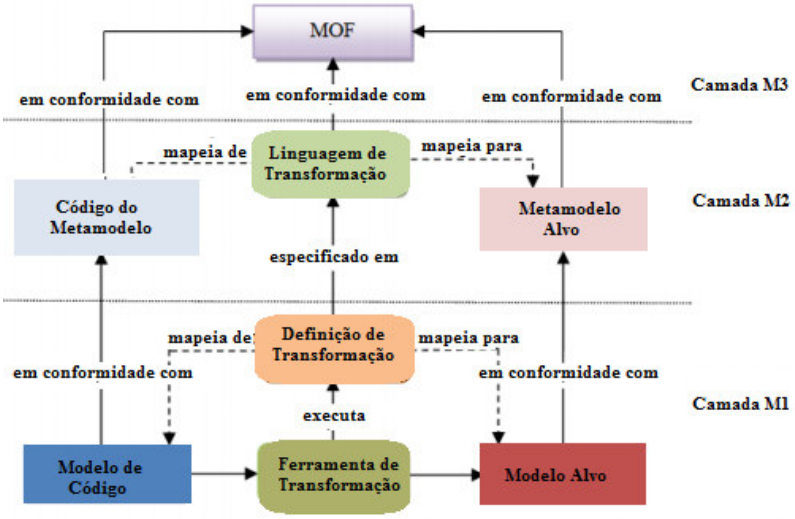
Fonte: Traduzido de Atnkison e Kuhne (2015).

A infraestrutura (Figura 2.13), segundo Atnkison e Kuhne (2015) possui quatro níveis de hierarquia de modelos, cada um podendo ser caracterizado como um exemplo do nível acima. Esses níveis são divididos da seguinte maneira:

- M0: camada de menor nível de abstração, contém os dados atuais dos objetos do *software* que será manipulado, sejam eles objetos de um sistema orientado a objetos ou registros em tabelas de banco de dados;
- M1: contém os modelos da aplicação, como diagramas de classes em UML;
- M2: mantém um modelo nas informações do M1, podendo ser considerado o metamodelo, por exemplo, o metamodelo da linguagem UML; e,
- M3: considerado o meta-metamodelo, mantém os dados da aplicação do M2, ou seja, define os elementos básicos para a criação de metamodelos.

Sharma e Sood (2011) definem outra forma de representar essa infraestrutura (Figura 2.14). Os autores definem que o padrão MOF é o principal artefato da MDA, uma vez que permite que as transformações ocorram entre diferentes metamodelos.

Figura 2.14: Modelo de transformação em MDA.



Fonte: Traduzido de Sharma e Sood (2011).

Observa-se que o PIM e o PSM podem ser especificados na camada M1 da arquitetura MOF, como é chamada por Sharma e Sood (2011). Ambos devem estar em conformidade com seus respectivos modelos a nível M2.

Conclui-se que um dos principais objetivos da MDA é facilitar a transformação de modelos em código, com o intuito de obter flexibilidade em longo prazo, além de interoperabilidade, portabilidade, e reutilização através da separação de conceitos arquitetônicos. Logo, a abordagem MDA para o desenvolvimento de *software* promete beneficiar os *stakeholders* mediante aumento da produtividade, melhoria do *software*, preservação dos custos e redução de tempo (SHARMA e SOOD, 2011).

2.5 Os Requisitos em MDD

Segundo Kovačević *et. al.* (2007), mudanças de requisitos sempre foram um grande problema na ES, e devido a isso, são necessários recursos capazes de identificar os impactos com o intuito de evitar maiores esforços de manutenção. Gerenciar as mudanças sofridas pelos requisitos em MDD é uma atividade complexa quando realizada de maneira manual. No contexto do MDD, existe uma variedade de métodos e técnicas que tem sido publicados na literatura. Entretanto, há poucos trabalhos que explicitam a importância dos requisitos nesse ambiente.

Os autores apresentam uma análise de como os critérios gerais da ER se adequam às técnicas do MDD. Algumas das características analisadas são:

- Evolução: em MDD, a evolução está relacionada com a possibilidade de alterar modelos determinados na fase de especificação de requisitos, com a opção de incluir ou remover requisitos existentes;
- Verificação: identifica se o modelo satisfaz as condições estabelecidas no início do processo de transformação. Trata-se de uma avaliação da conformidade e consistência do modelo. Nesse caso, a conformidade verifica se o modelo satisfaz as restrições impostas e a consistência checa se a informação dada a um modelo específico corresponde à informação dada a outro modelo nos casos em que os modelos interagem;
- Rastreabilidade: as transformações entre os modelos são sistemáticas e rigorosas. Devido a isso, o MDD deve fornecer informações intrínsecas básicas, que podem ajudar na rastreabilidade dos requisitos. Sendo assim, grande parte das decisões deve ser documentada entre cada transformação de modelos, que podem servir como entrada para a rastreabilidade; e,
- Ferramenta de Suporte: as ferramentas de suporte devem realizar transformações automáticas gerando modelos resultantes e o código fonte. Em nível de requisitos, deve ser criado um primeiro modelo com as informações necessárias que será a entrada para o início das transformações.

A variedade de artefatos existentes em MDD possui o propósito de produzir realizações que satisfaçam um determinado conjunto de requisitos, respeite as limitações de implementação e princípios gerais de *design*. Como no desenvolvimento de *software* tradicional, em MDD os requisitos também precisam ser monitorados e gerenciados durante as transformações. Além disso, ele deve ser capaz de se adaptar às mudanças dos artefatos ocasionadas pela alteração de requisitos já existentes, ou pela inclusão ou exclusão de requisitos.

2.6 Considerações Finais do Capítulo

Neste capítulo foram apresentados conceitos sobre as principais abordagens dirigidas a modelos, evolução e problemática. As abordagens levantadas foram MDE, MDD e MDA. Como visto, diversos autores sugerem uma grande mudança de paradigma em que os modelos passarão a ser o artefato principal do desenvolvimento de *software*. Dentre as abordagens

descritas, o MDD foi enfatizado pelo fato de possuir uma perspectiva do processo de desenvolvimento.

Nesse contexto, uma série de benefícios foram listados, como a melhoria na produtividade, manutenção, comunicação com interessados, adaptação, entre outros. No entanto, os benefícios prometidos pelo MDD ainda não são aproveitados de maneira completa, uma vez que ainda existem restrições quanto a sua utilização. Uma dessas restrições está relacionada à dificuldade de rastrear e monitorar requisitos durante todo o processo. O próximo capítulo apresenta uma revisão teórica da ER levantando seus principais processos e o impacto da rastreabilidade de requisitos em projetos de *software*.

CAPÍTULO 3

ENGENHARIA DE REQUISITOS

Antes de começar a projetar um sistema, se faz mister entender quais são os desejos dos clientes. Isso se deve ao fato de que o esforço do desenvolvimento pode ser parcial ou totalmente perdido quando o *software*, mesmo pronto, não atende às reais necessidades. Com isso, segundo Falbo (2012), os requisitos tornam-se a base de todo projeto, já que eles devem definir os interesses dos *stakeholders* e o que o sistema deve fazer a fim de atingir objetivos. De acordo com o autor, são os requisitos que definem o sucesso ou fracasso dos projetos.

Para Hull *et al.* (2002), no entanto, uma das atividades mais complexas da Engenharia de *Software* é compreender os requisitos de um problema. Para o autor, o maior desafio para esse entendimento é capturar as necessidades ou problemas de forma completa e sem equívocos. Isso ocorre porque as necessidades dos *stakeholders* podem ser muito variadas, não serem apresentadas de forma clara no início do projeto, limitadas a fatores fora de controle ou podem variar de acordo com o tempo de desenvolvimento. Logo, se não houver uma base sólida de gerência e manutenção dos requisitos, o projeto pode não obter êxito ao final.

A ER busca minimizar as dificuldades dos engenheiros de *software* para levantar os requisitos do sistema a ser desenvolvido por meio de atividades que levam a um entendimento do negócio, das necessidades do cliente e de como será a interação do usuário com o *software* (PRESSMAN, 2016). Esse capítulo visa apresentar os principais conceitos da ER, o que são requisitos, documentos de requisitos e os processos da ER.

3.1 Entendendo a Engenharia de Requisitos

Segundo Falbo (2012), o processo de levantar, analisar, documentar, gerenciar e controlar os requisitos é chamado de Engenharia de Requisitos. De acordo com o autor, a ER possui uma diversidade de definições e termos existentes.

Para entender a Engenharia de Requisitos, faz-se necessário o conhecimento do que é um requisito. Segundo Autumn (2004), os requisitos são as definições dos serviços e das

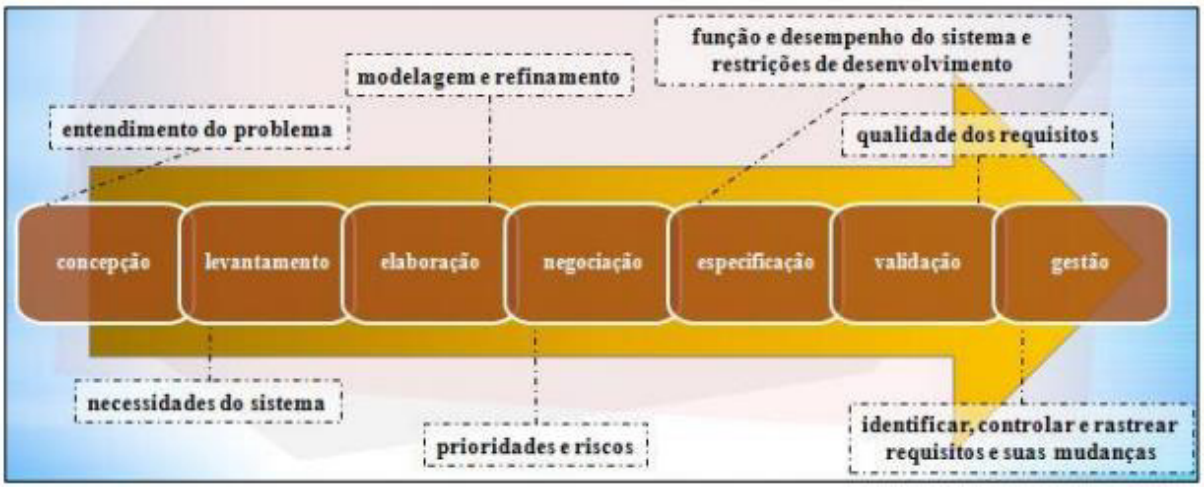
restrições que um *software* deve possuir. Sommerville (2007) faz a mesma afirmação, porém acrescentando que os requisitos devem atender às necessidades dos clientes e auxiliar na resolução de um problema específico.

O padrão 830-1998 do *Institute of Electrical and Electronics Engineers* (IEEE) define requisitos como uma condição necessária para resolver um problema ou alcançar um objetivo. Segundo Leite (1999), "requisitos são objetivos ou restrições estabelecidas por clientes e usuários de sistemas que definem as propriedades do sistema". Na visão de Mello e Meyer (2010), os requisitos são como uma coleção de sentenças que descrevem de maneira coesa todos os aspectos do sistema proposto.

3.2 Fases da Engenharia de Requisitos

Segundo Pressmann (2016), a ER oferece um mecanismo adequado para a compreensão do que o cliente precisa através de uma análise de necessidades, negociação de uma condição apropriada, definição de uma solução sem ambiguidades, bem como a validação, especificação e gerência de requisitos à medida que a codificação está sendo feita. Para isso, a ER define funções que devem ser seguidas e englobam o processo da ER (Figura 3.1).

Figura 3.1: Funções do processo de ER.



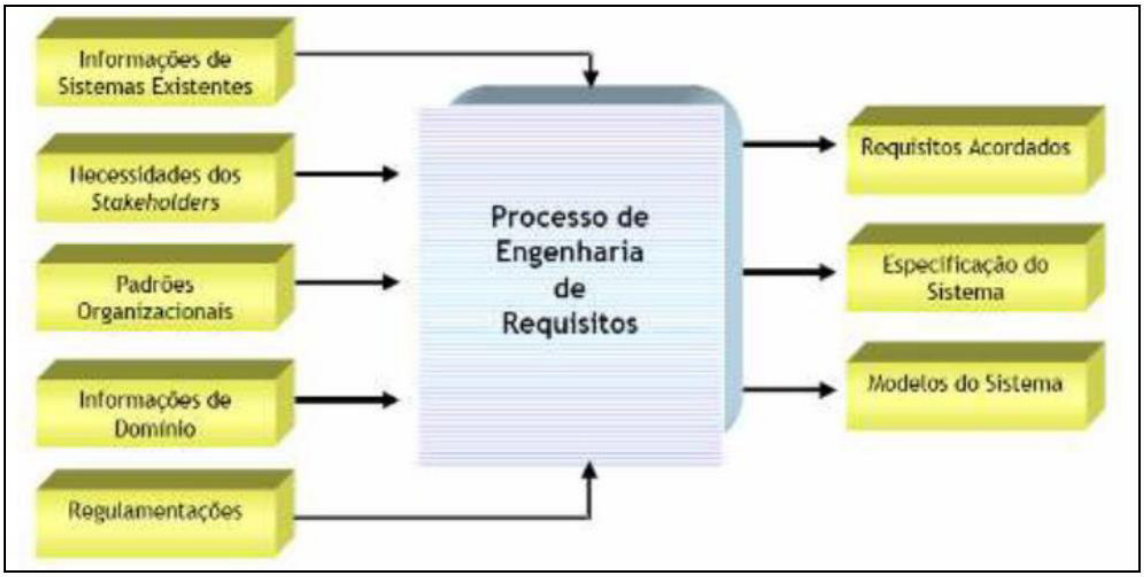
Fonte: (SANCHES *et al.*, 2014).

Para Sanches *et al*. (2014), essas funções são necessárias para a execução do processo da ER. As funções identificadas na figura podem ser descritas conforme Pressman (2016) da seguinte maneira:

- Concepção: por meio de uma série de questões livres de contexto, estabelece um entendimento básico do problema, os envolvidos, a natureza do *software* e a forma de comunicação entre o cliente e o desenvolvedor;
- Levantamento: identificar por meio de perguntas e questionários ao usuário qual o objetivo do sistema ou produto, o que necessita ser realizado, quais são as necessidades do negócio e como ele será utilizado;
- Elaboração: refinamento das informações levantadas nas fases anteriores com foco na elaboração de um modelo técnico apurado das funções, características e restrições. resulta em um modelo em que consta o domínio do problema em termos de informações, funcionalidades e comportamentos;
- Negociação: é a conciliação que ocorre junto aos clientes e usuários quanto ao que pode ou não ser realizado considerando os recursos necessários ao desenvolvimento do *software*;
- Especificação: é a descrição das funções e desempenho do *software* e as restrições que conduzirão seu desenvolvimento;
- Validação: avaliação do produto em termos de qualidade podendo ser feita pelos engenheiros, clientes, usuários e outros interessados que buscam erros de conteúdo ou interpretação; e,
- Gestão: conjunto de tarefas que possuem o objetivo de identificar, controlar e rastrear requisitos ao longo da vida do *software*.

Cada atividade de um processo de ER pode ou não resultar em um produto, o qual precisará ser mantido e estar apto à avaliação de qualidade. Para execução do processo da ER é preciso saber quais são as entradas e as saídas existentes no processo, as quais são propostas por Sommerville (2007) (Figura 3.2).

Figura 3.2: Entradas e saídas do processo de ER.



Fonte: (SOMMERVILLE, 2007).

As entradas apresentadas podem ser descritas da seguinte maneira:

- Informações sobre sistemas existentes: dados sobre as funcionalidades do *software* que será substituído ou de outros sistemas que irão interagir com o *software* a ser desenvolvido;
- Necessidades das partes envolvidas: informações sobre as necessidades e objetivos do *software* a ser desenvolvido;
- Padrões organizacionais: padrões utilizados pela equipe de desenvolvimento, gerência de qualidade, entre outros;
- Regulamentações: regras que se aplicam ao sistema; e,
- 'Informações do domínio: dados gerais sobre o domínio do sistema.

Ainda segundo Sommerville (2007), as saídas são:

- Acordos sobre os requisitos: descrição de todos os requisitos em concordância com todos os envolvidos;
- Especificação do sistema: detalhamento das informações e funcionalidades do sistema; e,
- Modelos do sistema: conjunto de modelos que descrevem o sistema.

Os processos da ER podem variar a depender do domínio da aplicação, das pessoas envolvidas e da organização ao definir os requisitos. Existem, porém, quatro processos comuns entre os existentes. Segundo Franceto (2005), essas etapas compreendem: Elicitação

dos Requisitos, Análise e Negociação dos Requisitos, Documentação/Especificação dos Requisitos, Validação dos Requisitos e Gerência de Requisitos.

A elicitação de requisitos refere-se ao levantamento do conhecimento relacionado ao problema a ser resolvido. É nesse momento que o cliente e o desenvolvedor analisam as necessidades do usuário e as restrições ou validações que o sistema deverá ter. Para Kotonya e Sommerville (1998), nessa fase são avaliados os processos, problemas e dificuldades, falhas nos *softwares* atuais, restrições impostas pelo negócio e possíveis melhorias. Para isso, é necessário estudar a organização, analisar os processos de negócio e ponderar o contexto em que a empresa está inserida no mercado.

Após coletar todos os requisitos, é necessário realizar a análise e negociação dos requisitos em acordo com os *stakeholders*. O objetivo dessa fase é gerar descrições dos requisitos de maneira que eles possam ser facilmente compreendidos. Esses requisitos devem ser discutidos, negociados e priorizados juntamente com os interessados do sistema. O objeto de retorno dessa fase são os requisitos completos e organizados em ordem de prioridade.

De acordo com Pressman (2016), a análise de requisitos é um ponto crucial para a revisão e consequentemente para a coerência da especificação. Ainda segundo o autor, essa etapa auxilia o analista no entendimento da informação e no comportamento do sistema. Para Falbo (2012) *apud* Pfleeger (2004), essa fase fornece uma especificação que oriente os desenvolvedores nas etapas de desenvolvimento, inclusive no projeto e testes do sistema. Além disso, ela provem um melhor entendimento e concordância entre interessados e desenvolvedores sobre como o sistema deve se comportar. Esta é uma fase considerada conceitual, ou seja, possui foco somente na dimensão do problema, porém, os recursos técnicos não são detalhados.

Os requisitos levantados e os modelos obtidos nas fases anteriores devem ser expostos e descritos na fase de Documentação de Requisitos. Portanto, para Falbo (2012), a documentação é uma atividade que oficializa os resultados da ER. Nesse processo ocorre a criação de um documento que apresente todos os requisitos do sistema de *software* de maneira coerente. Todos os envolvidos devem entender esse documento, uma vez que ele deverá servir como contrato entre o cliente e o desenvolvedor.

Para Falbo (2012), a documentação permite a melhor comunicação dos requisitos e facilita os processos de desenvolvimento, já que auxilia os programadores e analistas e evita o retrabalho em fases posteriores. O autor aponta alguns benefícios da documentação, como a facilidade de passagem de conhecimento para outros usuários, geração de uma base

consolidada que permite estimar fatos e possíveis necessidades do sistema, além de servir como apoio para novas funcionalidades e ajudar no controle e manutenção do sistema.

Após a fase de documentação, a validação e verificação são de suma importância para o processo de desenvolvimento de *software*. Falbo (2012) *apud* Rocha *et al.* (2001) afirmam que os requisitos devem ser constante e cuidadosamente avaliados, fazendo com o que a documentação seja submetida à validação e à verificação.

Validação e verificação são atividades diferentes. Para Falbo (2012) *apud* Rocha *et al.* (2001), a verificação tem como objetivo garantir que o sistema está sendo desenvolvido de forma correta, assegurando que os padrões da organização estão sendo aplicados e que os artefatos estão atendendo os requisitos instituídos. Já a validação garante que os requisitos e o sistema atendem o uso sugerido e que o *software* que está sendo construído é o correto. Além disso, a validação garante que o sistema faz o que foi solicitado e atende as demandas dos usuários.

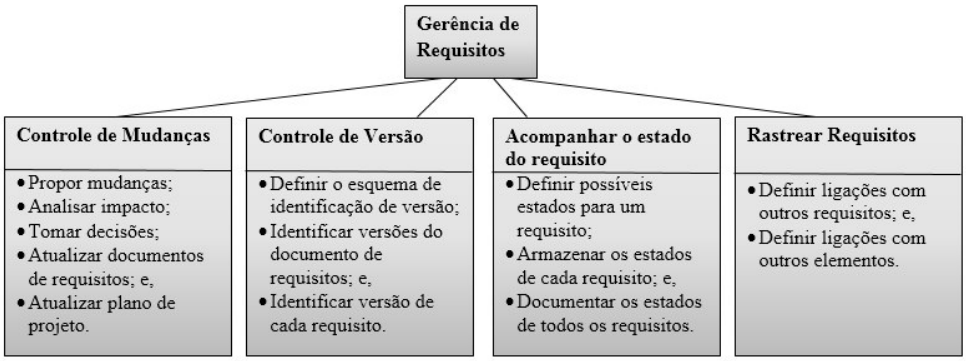
3.2.1 Gerência dos Requisitos

Os requisitos durante todo o processo de *software* sofrem alterações que podem ocorrer por diversos motivos, como problemas no sistema, omissões de informações, mudanças de prioridade do cliente, variações de cronograma, entre outros. Sayão e Leite (2005) afirmam que essas mudanças necessitam ser acompanhadas para que os componentes afetados possam ser corrigidos.

Nesse sentido, a gerência de requisitos inclui atividades que auxiliam a equipe a controlar, identificar, rastrear e gerenciar mudanças durante o ciclo de vida do *software*. Para Kotonya e Sommerville (1998), os principais objetivos desse processo são gerenciar as alterações, os relacionamentos e as dependências entre os requisitos. Esse processo equivale ao planejamento do controle da elicitação, análise e negociação, documentação e validação dos requisitos englobando todas as fases anteriores da ER.

Wiegers (2004) divide o processo de gerência nas atividades de: controle de mudanças (registro ou análise de modificações ou inclusões de requisitos), controle de versões (dos diferentes artefatos), acompanhamento do estado de requisitos e rastreabilidade de requisitos (Figura 3.3).

Figura 3.3: Atividades da Gerência de Requisitos.



Fonte: Wiegers (2004).

Cada atividade tem seu grau de importância na ER. Entretanto, os requisitos não podem ser efetivamente gerenciados sem a rastreabilidade de requisitos, etapa que apoia diversos tipos de análise e que pode indicar se um sistema atende ao que foi especificado.

3.3 Rastreabilidade e Monitoramento de Requisitos

A rastreabilidade de requisitos é uma das etapas mais importantes da gerência de requisitos e também da ER. Segundo Sayão e Leite (2005), os requisitos não podem ser gerenciados de forma efetiva sem a rastreabilidade. Essa etapa trata do acompanhamento do requisito durante o ciclo de vida do software. Para eles, a rastreabilidade auxilia gerentes e desenvolvedores em diversas situações, como na identificação de requisitos ainda não alocados ou implementados, na origem dos conflitos entre requisitos, auxílio na etapa de verificação e validação com o apontamento de componentes que geraram erros, indicação de componentes afetados devido às mudanças de requisitos permitindo uma análise de impacto, previsões de alterações em cronogramas, já que é possível saber quais os impactos de mudanças nos requisitos, dentre outras situações. A próxima seção apresenta os conceitos básicos associados à rastreabilidade.

3.3.1 Rastreabilidade

A rastreabilidade é definida de forma genérica como o grau das relações existentes entre os artefatos resultantes do processo de desenvolvimento de *software*. Essa etapa está relacionada a manter ligações entre os artefatos, isto é, fornecer meios para determinar as

relações e dependências entre os artefatos que apoiam algumas atividades de ES, tais como o impacto das alterações, análise e manutenção (IEEE, 1990).

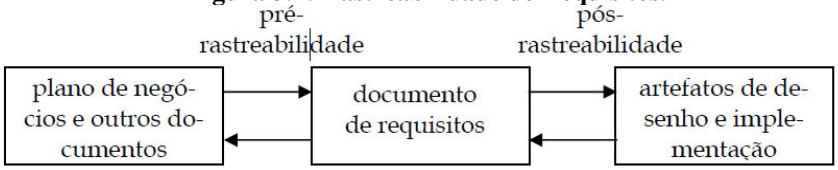
No contexto da ER, Edwards (1991) *apud* Sayão e Leite (2005) definem rastreabilidade como uma técnica para fornecer relacionamentos entre requisitos, artefatos de requisitos, arquitetura e implementação final do *software*. Para os autores, a rastreabilidade ajuda no entendimento e compreensão desses relacionamentos, bem como permite aos projetistas concluir se o produto atende aos requisitos.

Palmer (1997) afirma que com isso tem-se o embasamento para uma avaliação de todo o projeto, sendo que outros componentes e artefatos afetados podem ser encontrados e alterados, se necessário. Para isso é preciso que haja relacionamentos entre os requisitos e entre requisitos e outros itens do processo de software.

A norma IEEE 830-1984 afirma que "uma especificação de requisitos é rastreável se (i) a origem de cada um dos seus requisitos é clara e se (ii) facilita a referência de cada requisito no desenvolvimento futuro ou na documentação de apoio".

O conceito mais comumente utilizado é definido por Gotel e Finkelstein (1994), que afirmam que a rastreabilidade de requisitos refere-se à capacidade de descrever e seguir a vida de um requisito em duas direções: para frente e para trás, ou, segundo Sayão e Leite (2005), pré-rastreabilidade e pós-rastreabilidade respectivamente (Figura 3.4).

Figura 3.4: Rastreabilidade de Requisitos.



Fonte: Sayão e Leite (2005).

Na pré-rastreabilidade é feita a documentação do contexto a partir do qual insurgem os requisitos e a pós-rastreabilidade faz um vínculo dos requisitos com o desenho do sistema e sua implementação. Ainda para Sayão e Leite (2005), é por meio da rastreabilidade que é possível descobrir a evolução de todas as características do sistema, bem como a identificação dos impactos provenientes das mudanças nos requisitos e ainda o rastreamento das permissões de mudanças de qualquer artefato.

Segundo Neto (2011), a pré-rastreabilidade acontece quando os requisitos são alterados e é necessário investigar qual o impacto da mudança. Nesse caso, podem-se obter todos os procedimentos de testes relacionados para a decisão de quais destes devem ser

acrescidos, alterados ou excluídos para que os processos relacionados a testes estejam conforme o requisito modificado. A pós-rastreabilidade ocorre quando, por exemplo, existe uma modificação e é necessário compreendê-la, com a investigação das informações utilizadas para elicitar o requisito modificado. É possível saber quem tem interesse no requisito ou os documentos pelos quais o requisito foi extraído, bem como em quais departamentos da organização o requisito está relacionado.

3.3.2 Rastro (*Trace*)

Segundo Leal (2011), o rastro é o principal elemento do trabalho da rastreabilidade. Para o autor, "artefatos são criados e modificados como resultado de tarefas executadas durante o processo de desenvolvimento do software. Essas tarefas deixam rastros que ligam os artefatos trabalhados". Em um sistema de gestão de configurações, por exemplo, existem registros de quem trabalhou no artefato, quando trabalhou e quais partes do artefato foram alteradas. Essas informações são características de um rastro que resultou de uma tarefa executada.

Ainda segundo Leal (2011), um rastro pode ser uma indicação ou evidência do que aconteceu ou do que existiu. No contexto da ES, o que aconteceu, tanto pode ser considerado um evento ocorrido durante o ciclo de desenvolvimento, como pode ser uma tarefa realizada pertencente ao projeto, ou ainda um acontecimento externo. Para o autor, os rastros são tratados como instâncias de tipos que representam algumas dimensões de informações de rastreabilidade definidas por Ramesh e Jarke (2001):

- O que: que informação é representada? É um requisito, uma suposição, uma restrição ambiental? E em que outras informações se baseia?
- Quem: quais são as partes interessadas que executam papeis na criação e manutenção dos rastros? Quais outras partes interessadas estiveram ou estão envolvidas no processo?
- Como: como a informação é apresentada? Documentação formal ou texto informal, gráficos, gravações de áudio ou vídeo? Como se relacionam com outros componentes de rastreabilidade?
- Onde: onde a informação é representada? Qual é a fonte da informação?

- Porque: qual o motivo de certo elemento ser criado ou evoluído? Qual é a lógica por trás do artefato? Existiam outras alternativas? Por que essa alternativa foi escolhida?
- Quando: quando a informação foi capturada, criada ou evoluída?

Algumas dessas perguntas são de fácil resposta, por exemplo, como, quem e quando. Entretanto, as demais somente podem ser respondidas depois da criação ou alteração do artefato e somente pelos *stakeholders* envolvidos. A documentação dessas informações é importante, uma vez que corre o risco de esquecimento caso elas permaneçam somente na mente das partes interessadas.

3.3.3 Ligações de Rastreabilidade

As ligações ou elos de rastreabilidade são relacionamentos que documentam dependências, influências, casualidade, entre outras informações existentes entre os artefatos. Uma ligação de rastreabilidade possui algumas características, como artefato de origem, artefato de destino e tipo de ligação (Figura 3.5) (GENVIGIR e VIJAYKUMAR, 2008).

Figura 3.5: Elementos básicos de um elo.

Fonte: Genvigir e Vijaykumar (2008).

Existem dois tipos de classificação de ligações de rastreabilidade definidas por Gotel e Finkelstein (1994), a pré-especificação de requisitos (pré-RS) e a pós-especificação de requisitos (pós-RS). Na pré-RS são definidas ligações durante a fase de concepção do *software,* e na pós-RS ocorre a criação de ligações durante o projeto e implementação.

Existem propostas que categorizam as ligações de rastreabilidade em diferentes tipos. Ramesh e Jarke (2001), propuseram:

- Satisfação e Dependência: grupo que descreve propriedades e relacionamentos com objetos; e,
- Evolução e Fundamentação: grupo que se refere ao processo, o qual pode ser capturado se observado o histórico das ações.

Já Toranzo *et al.* (2002) categorizaram da seguinte forma:

- Satisfação: o elemento de origem satisfaz o de destino, como por exemplo, um componente de *software* satisfaz o objeto;

- Recurso: o elemento de origem é considerado um recurso do destino, como por exemplo, uma proposta de mudança utiliza requisitos de sistema como recurso de informação;
- Responsabilidade: indica a participação, ação ou responsabilidade de integrantes sobre os itens gerados;
- Representação: modelagem ou representação visual dos requisitos em outras linguagens;
- Alocação: elementos de origem reservados para serem utilizados por elementos de destino; e,
- Agregação: composição entre elementos.

As classificações dos elos podem ser determinadas utilizando como base atributos ou propriedades ou a aplicação dessas ligações no processo de desenvolvimento. A existência das ligações de rastreabilidade torna possível identificar as origens de cada requisito. Outros autores apresentam formas de categorizar os elos, entretanto, essas são as pesquisas mais relevantes.

3.3.4 Matriz de Rastreabilidade

O rastreamento dos requisitos pode ser definido através de um conjunto de elos e ligações entre os requisitos em si, entre os requisitos e suas fontes, entre os requisitos e as razões básicas de suas hipóteses e os componentes que os implementam (OLIVEIRA e PAIVA, 2009). Com o objetivo de facilitar o rastreio dos requisitos são utilizadas matrizes de rastreamento que criam as ligações e associações entre os requisitos e suas dependências. Um exemplo de matriz de rastreabilidade pode ser visualizado no Quadro 3.1.

Quadro 3.1: Matriz de Rastreabilidade.

	Req 1	Req 2	Req 3	Req 4	Req 5
Req 1			X		X
Req 2	X			X	
Req 3		X			X
Req 4		X		X	
Req 5			X	X	X

Pode-se observar, por exemplo, que o requisito Req1 é dependente de Req3 e Req5, como o requisito Req2 depende de Req1 e Req4. Nesse caso, se por algum motivo existir a necessidade de se alterar o requisito Req4, os requisitos Req2 e Req5 serão afetados. A

existência dos elos de rastreabilidade possibilita identificar as origens das funcionalidades apresentadas no sistema.

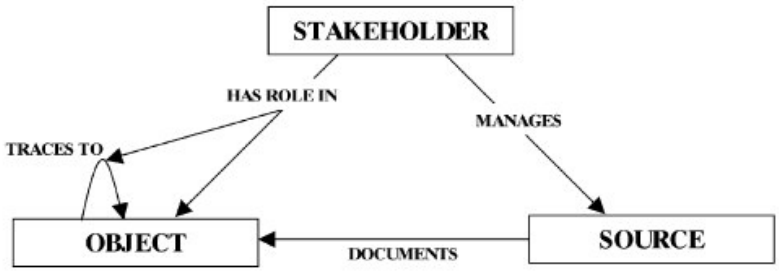
3.4 MMRs de Requisitos Encontrados na Literatura

As informações relacionadas ao rastreamento de requisitos podem ser representadas através de metamodelos de rastreabilidade (MMR), que são genéricos e podem ser aplicados em diversas situações. Os MMRs buscam maneiras de determinar as características de um rastro e seus relacionamentos. Esta seção apresenta três metamodelos encontrados na literatura que buscam definir propriedades de um rastro.

3.4.1 Metamodelo Proposto por Ramesh e Jark (2001)

Um dos primeiros metamodelos de rastreabilidade foi proposto por Ramesh e Jark (2001) com base em estudos empíricos. Os autores propuseram um metamodelo (Figura 3.6) que satisfaz as dimensões (o que, quem, como, onde, quando e porque) de rastreabilidade.

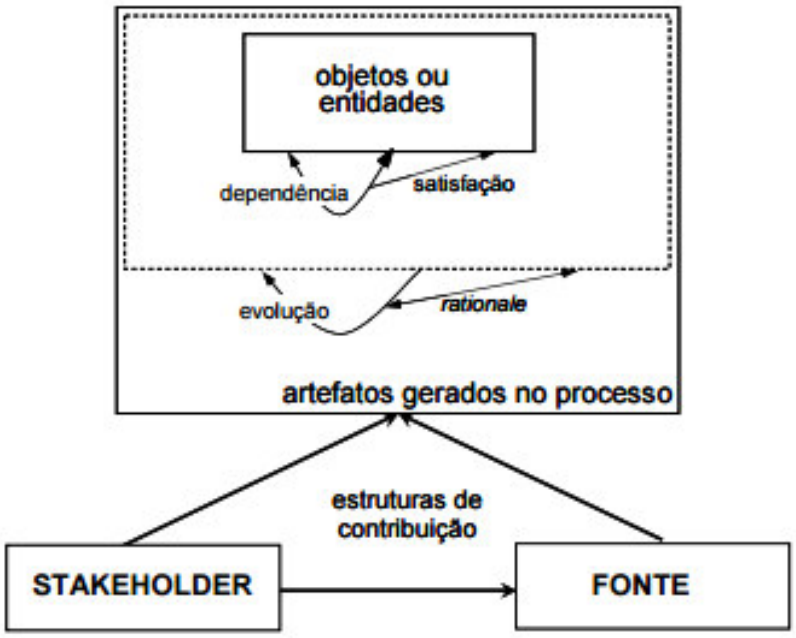
Figura 3.6: Metamodelo proposto por Ramesh e Jark (2001).



Fonte: Ramesh e Jark (2001).

É possível observar que este metamodelo apresenta informações relacionadas a objetos (*Object*), fontes (*Source*) e interessados (*Stakeholder*). A fonte refere-se a origem dos requisitos, como legislação, atas de reuniões etc.; os interessados são pessoas envolvidas e que possuem determinado interesse na rastreabilidade; e, os objetos referem-se aos objetos conceituais relacionados ao produto ou artefatos gerados durante o desenvolvimento (SAYÃO e LEITE, 2005). Uma visão geral do metamodelo pode ser vista na Figura 3.7.

Figura 3.7: Visão do metamodelo proposto por Ramesh e Jark (2001).



Fonte: Sayão e Leite (2005).

Na visão de Sayão e Leite (2005), também são abordados os elos de rastreabilidade agrupados por duas categorias: produtos e processos. Os elos relacionados aos produtos descrevem os objetos e podem ser divididos em satisfação e dependência, enquanto que os associados aos processos, referem-se ao histórico de ações executadas e são divididos em elos de evolução e *rationale.*

Os elos de satisfação asseguram que os requisitos irão atender às necessidades do cliente enquanto que os de evolução registram relacionamentos entre objetos existentes e novos ou modificados. Os elos de *rationale* representam a motivação referentes aos objetos existentes e a justificativa pela qual ocorreu a criação ou modificação de um determinado objeto. E, por fim, os elos de dependência buscam o gerenciamento de dependências entre objetos, composição e hierarquia sendo úteis no apoio de análise de impactos de alterações e outros eventos.

Segundo Ramesh e Jark (2001), o metamodelo abrange as seguintes dimensões de rastreabilidade:

- Que informação é representada? A metaclasse *Object* representa as entradas e os resultados do processo de desenvolvimento. Os requisitos, pressupostos, projetos, a fundamentação, as alternativas, entre outros, são exemplos do que podem se tornar instâncias de *Object*. Essas informações representam as principais informações de rastreabilidade que são mantidas durante as várias

fases do ciclo de vida. A rastreabilidade entre os objetos é representada por meio das ligações *trace-to*.

- Quem são os interessados na rastreabilidade? A metaclasse *Stakeholders* representa os agentes envolvidos no desenvolvimento do sistema e gerenciamento do ciclo de atividades. Estes, podem ser gerentes de projetos, analistas de sistemas, *designer*, entre outros. Os *stakeholders* atuam em diferentes funções (*Roles*), que são as capacidades em estabelecer e usar os objetos e *links* de rastreabilidade.
- Onde é representada? Todos os objetos são documentados por *Sources.* Estes, podem ser uma média física, documentos de especificação de requisitos, de *design*, gravações, chamadas telefônicas, entre outros. Os *Sources* são gerenciados pelos *stakeholders.* Essa informação é representada por *Manages.*
- Como é representada? As informações podem ser os documentos de texto, gráficos, documentos de *design,* entre outros, representadas em *Source*.
- Por que certo objeto foi criado, modificado ou evoluído? As informações necessárias para responder esta pergunta podem ser representadas através de uma especialização da metaclasse *Object.* Os modelos mais completos podem ser representados como especialização de *Object-trace-Object.*
- Quando esta informação foi capturada, modificada ou evoluída? Essa informação pode ser armazenada em um atributo em qualquer metaclasse do metamodelo.

Apesar do metamodelo tratar das dimensões da rastreabilidade, ele não permite que elas sejam representadas em um único rastro. De acordo com Júnior (2005), nele, os rastros não são elementos únicos com cinco dimensões. Na verdade, algumas dimensões devem ser representadas através de instâncias de tipos que representam algumas das dimensões. Uma ligação que indica que um componente satisfaz um requisito, por exemplo, deve ser representada com os elementos "Componente" e "Requisito" e a ligação por meio do "*Trace to*".

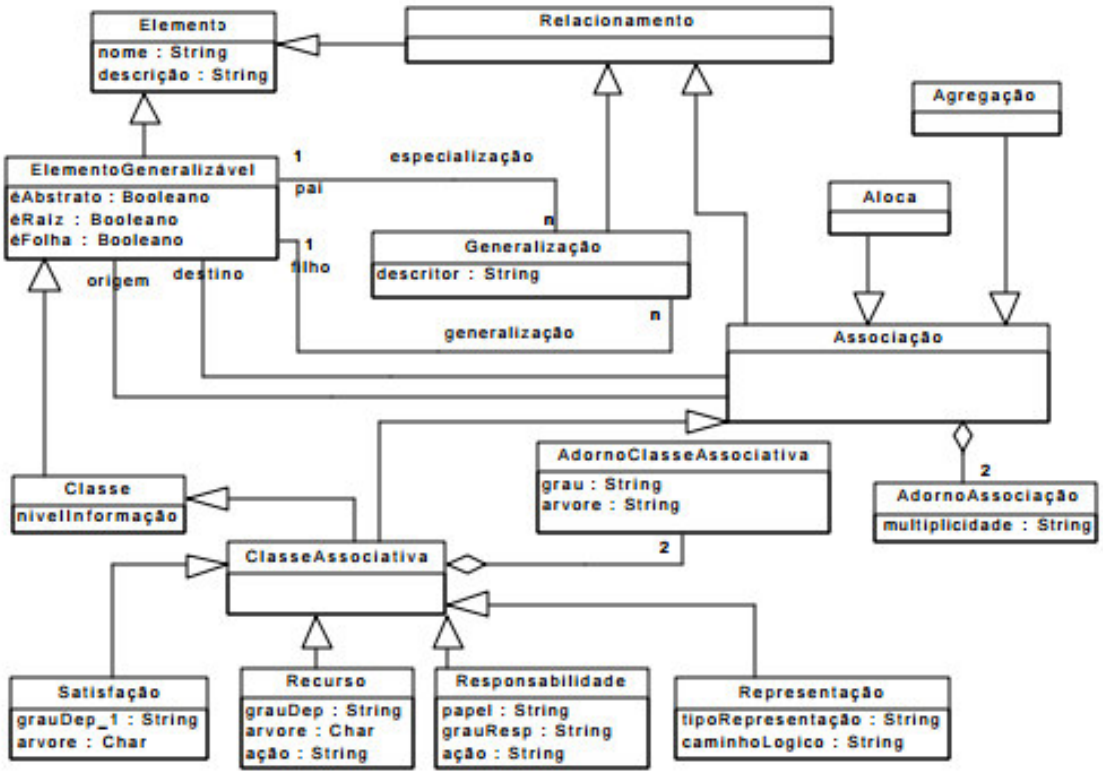
Esse rastro então, satisfaz três dimensões, que são: "Quem são as partes interessadas?", "Onde está representada?" e "Que informação é representada?", entretanto, as demais dimensões só podem ser definidas através da especialização da metaclasse *Object*. Além disso, este metamodelo foi utilizado para explorar diferentes formas e tipos de rastreabilidade, entretanto, ele não é um metamodelo formal no contexto de MDD.

3.4.2 Metamodelo Proposto por Toranzo (2002)

O metamodelo proposto por Toranzo (2002) (Figura 3.8) foi fundamentado no MOF (OMG, 2000). Seu objetivo é facilitar a construção de um diagrama de classes que deve expressar informações de rastreamento de requisitos. O metamodelo utiliza quatro classificações de informações de rastreamento propostas pelo autor, sendo elas:

- Ambiental: incorpora informações no contexto ambiental em que a organização está inserida e que podem afetar o sistema que está sendo desenvolvido;
- Organizacional: une dados referentes à organização que podem impactar nos requisitos de sistema;
- Gerencial: congrega informações que ligam as tarefas aos requisitos e podem auxiliar na gerência de tarefas; e,
- Desenvolvimento: reúne dados acerca dos artefatos do projeto gerados durante o processo de desenvolvimento, que podem ser documento de requisitos, diagramas, programas, entre outros.

Figura 3.8: Metamodelo proposto por Toranzo (2002).



Fonte: Sayão e Leite (2005).

A metaclasse base do metamodelo é "Elemento". A subclasse "ElementoGeneralizável" classifica e categoriza as instâncias de "Elemento" e "Relacionamento", e possui as informações das relações entre os elementos. As instâncias de "ElementoGeneralizável" se relacionam por meio da "Associação" ou "Generalização". Os tipos de relacionamentos propostos e utilizados no metamodelo estão descritos no Quadro 3.2.

Quadro 3.2: Dicionário dos tipos de relacionamentos para rastrear requisitos.

TIPO DE ASSOCIAÇÃO	DESCRIÇÃO
Generalização	Um relacionamento entre um elemento geral e um elemento mais específico. O elemento mais específico é complemente consistente com o elemento geral e contém informações adicionais.
Aloca	Uma associação unidirecional de uma classe, que representa requisitos funcionais, para uma outra classe que representa um subsistema.
Agregação	Uma associação que modela um relacionamento todo-parte entre uma classe, denominado todo, e uma ou mais classes que são denominadas partes.
Satisfação	Uma associação entre duas classes denominadas classe origem e classe destino. A associação é desenhada da classe origem para a classe destino. A associação satisfação é uma relação de dependência que especifica que a classe origem tem uma dependência de realização com a classe destino. O termo realização é usado para expressar que algo deverá ser realizado ou feito sobre as instâncias da classe destino para satisfazer as instâncias da classe origem com as quais estão conectadas.
Recurso	Uma associação que especifica que uma classe possui uma dependência de informação ou física com uma outra classe.
Responsabilidade	Uma associação que visa capturar a participação, responsabilidade e ação das pessoas sobre artefatos ou elementos do processo de software.
Representação	Uma associação que é usada para mapear um elemento em um outro elemento.

Fonte: Toranzo (2002).

As seis dimensões de rastreabilidade não estão explícitas no metamodelo. Entretanto, as seguintes podem ser identificadas:

- Que informação é representada? Pode ser identificada através da metaclasse *Elemento*, a qual é formada pelos atributos *nome* e *descrição*.
- Quem são os interessados na rastreabilidade? Representada pela meta-associação chamada de *Responsabilidade.* Esta busca obter informações dos *stakeholders* e suas responsabilidades e ações em relação aos artefatos ou elementos.
- Como é representada? As informações podem ser representadas através dos tipos de associação entre os elementos, da multiplicidade, grau de dependência, dentre outros componentes presentes no metamodelo.

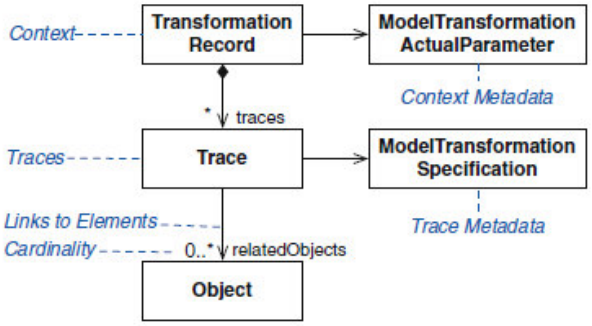
O metamodelo proposto por Toranzo (2002) contribui por focar em aspectos gerenciais do projeto. Um exemplo disso é a criação do elo "Responsabilidade" que tem o objetivo de registrar as responsabilidades das pessoas sobre os artefatos ou elementos. Esse é um dos fatores importantes, uma vez que a ER está ligada à Gerência de Projetos e à demanda de se conhecer os envolvidos. Assim como o metamodelo proposto por Ramesh e Jark (2001),

ele não trata dos aspectos relacionados ao MDD. Além disso, somente três dimensões de rastreabilidade podem ser identificadas no metamodelo: o que, quem e como.

3.4.3 Metamodelo Proposto por Winkler e Pilgrim (2009)

O metamodelo de rastreabilidade proposto por Winkler e Pilgrim (2009) (Figura 3.9) foi baseado em propriedades comuns aos principais metamodelos da MDE. O objetivo é ilustrar diferentes características de modelos de rastreamento.

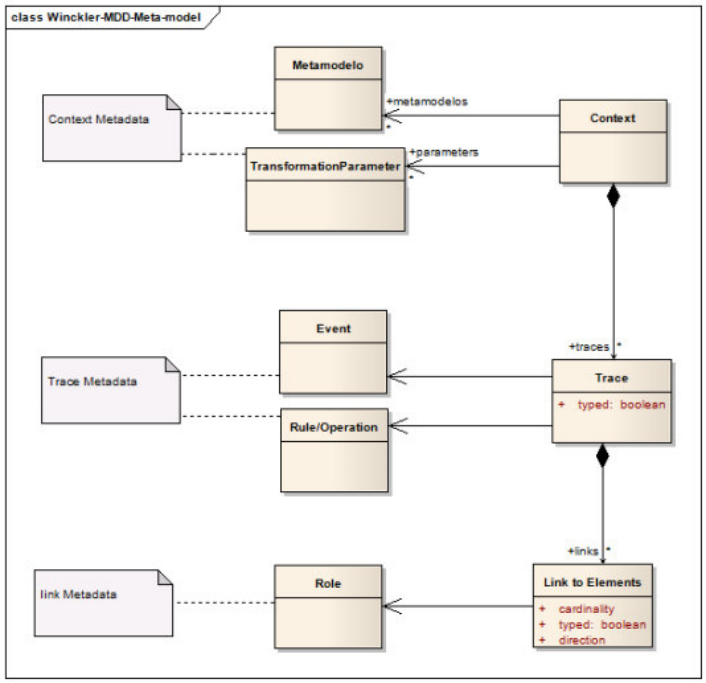
Figura 3.9: Metamodelo proposto por Winkler e Pilgrim (2009).



Fonte: Winkler e Pilgrim (2009).

A Figura 3.10 ilustra uma versão do metamodelo em XML segundo o ponto de vista de Júnior (2011).

Figura 3.10: Metamodelo Winkler e Pilgrim em XML.



Fonte: Júnior (2011).

Um *Context* é utilizado para armazenar determinados metadados, os valores de parâmetros de transformação, definir se a transformação foi automática ou se foi utilizada para criar índices, bem como, armazenar traços. Como o *Context*, o rastro *Trace* pode possuir um metadado, que pode ser uma regra ou uma operação que criou o rastro. A característica mais importante de um rastro são os *links* que ligam os elementos do modelo ou artefatos. Os tipos de ligações para rastrear requisitos definidos por Winkler e Pilgrim (2009) são:

- Dependência: *link* entre dois artefatos *e1* e *e2*, em que *e2* se baseia na existência de *e1* ou que as mudanças em *e1* resultarão em mudanças em *e2*.
- Refinamento: são utilizadas hierarquias de abstrações que descrevem como os artefatos complexos são divididos em partes menores, mais concretos ou mais gerenciáveis;
- Evolução: *links* são utilizados quando um artefato substitui outro, como por exemplo, quando uma nova versão substitui uma mais antiga ou se um conjunto de requisitos se sobrepõe com o intuito de elaborar um novo conjunto de requisitos;
- Satisfação: identifica se um artefato criado está em conformidade com os elementos e se eles satisfazem um ou mais requisitos;
- Sobreposição: Descreve quando dois artefatos possuem características ou aspectos comuns;
- Conflito: expressam um conflito existente em dois artefatos;
- Racionalização: utilizado para transportar informações sobre decisões e documentar a lógica envolvida na criação dos artefatos; e,
- Contribuição: descreve várias ligações entre artefatos e partes interessadas.

As dimensões identificadas no metamodelo são:

- Onde é representada? As informações de rastreabilidade são armazenadas no *Context*, através da metaclasse denominada *TransformationRecorder*.
- Como é representada? As informações são representadas através de suas ligações, que são os rastros ou *Traces.*
- Por que certo objeto foi criado, modificado ou evoluído? A metaclasse *Context* também pode responder essa pergunta, uma vez que ela armazena a proposta e a definição do modelo.

Os modelos de rastreabilidade são armazenados de forma separada e os *links* para os elementos são unidirecionais enquanto que as ligações entre modelos podem ser bidirecionais, com o objetivo de manter os modelos ou artefatos ligados de forma independente.

3.4.4 Análise dos Metamodelos Encontrados

Os três metamodelos analisados neste trabalho possuem algumas semelhanças e diferenças. Com relação aos elos de rastreabilidade, o único elo comum aos três metamodelos é o de satisfação. Em Ramesh e Jark (2001), a satisfação refere-se a atender as necessidades do cliente, enquanto que, em Toranzo (2002), ela é utilizada para indicar que algo deverá ser realizado sobre as instâncias da classe destino para satisfazer instâncias da classe origem com as quais estão conectadas. Já para Winkler e Pilgrim (2009), o termo indica se um artefato está em conformidade com os elementos e se eles satisfazem os requisitos.

O metamodelo de Toranzo (2002) possui foco nos aspectos gerenciais de desenvolvimento do projeto. Para tal, o autor utiliza o elo de responsabilidade. Ramesh e Jarke (2001) diferenciam-se de Toranzo (2002) porque buscam registrar tanto a evolução dos objetos como a motivação para modificações, abrangendo algumas dimensões da rastreabilidade. Já o diferencial do metamodelo de Winkler e Pilgrim (2009) inclui tratar os aspectos da MDE, as transformações de requisitos e os diferentes modelos de transformação.

Spinoza (2006) coloca que um metamodelo ideal para a rastreabilidade no ambiente de MDD deve compreender um conjunto de tipos de elos de rastreabilidade e o significado deles, um conjunto de informações que indicam por exemplo, os *stakeholders*, e respondem a todas as perguntas dos rastros e a especificação de um esquema que envolve regras de qualidade para a rastreabilidade de um respectivo projeto.

Os três metamodelos apresentados neste capítulo contribuem com a ER ao abrangerem aspectos básicos de rastreabilidade, como as dimensões e os elos. Entretanto, nenhum deles trata todas as questões e elos que os rastros devem responder.

3.5 Rastreabilidade de Requisitos em MDD: Análise dos resultados da RSL

Na última década foram propostos vários trabalhos que visam solucionar muitos dos problemas relacionados a rastreabilidade de requisitos. Poucos, porém, possuem foco ou

tratam de MDD. Esta seção faz uma avaliação dos trabalhos encontrados na RSL apresentada no Capítulo 1 desta pesquisa.

Todos os trabalhos avaliados destacam problemas e restrições quanto à rastreabilidade de requisitos em MDD. Alguns deles sugerem soluções, como *frameworks* e metodologia, descritos a seguir.

3.5.1 *Framework versus* Metodologia

Segundo Draffin (2007), o *framework* é uma estrutura lógica para classificação e organização de sistemas complexos. Pode ser uma imagem ou um modelo que orienta o entendimento do que deve ser produzido, como os artefatos, embora não mostre como fazê-lo. Por exemplo, em uma estrutura em cascata, a análise é seu primeiro estágio de desenvolvimento de *software*, mas o *framework* não diz como essa análise deve ser feita, ou seja, mesmo detalhando qual modelo deve ser utilizado, ele não informará como preencher tal modelo, dando a flexibilidade de escolha. Na visão de Rodrigues (2012), o *framework* pode ser um arcabouço que serve para associação de processos, métodos e técnicas, e que possui hipóteses, conceitos e práticas que norteiam sua execução.

Já a metodologia pode ser definida como o estudo dos métodos científicos cuja finalidade é aperfeiçoar procedimentos e critérios utilizados em uma pesquisa. A metodologia preocupa-se com a validade do caminho escolhido para se alcançar um resultado indo além da descrição dos procedimentos. Metodologia é diferente de método, que pode ser considerado como o meio ou caminho para se chegar a um objetivo (HEGENBERG, 1976) *apud* (THÉOPHILO e INDÍCIBUS, 2001).

Sendo assim, de forma comparativa, em uma metodologia, todos os processos são conhecidos e definidos de maneira prévia. Tratando-se de *framework*, existe a liberdade de escolher como a ação deve ser realizada dentro das necessidades e possibilidades, não sendo obrigatório seguir um processo prescrito.

3.5.2 Resultados da RSL

O Quadro 3.3 apresenta uma análise comparativa dos artigos, levando em consideração quais são os tipos de proposta dos autores, se eles mencionam ou sugerem ferramentas de apoio e quais ferramentas são listadas para a rastreabilidade.

Quadro 3.3: Análise comparativa dos artigos.

AUTORES	TIPO DE PROPOSTA	FERRAMENTA DE APOIO
Almeida *et al.* (2006)	*Framework*	Não utiliza.
Valderas e Pelechano (2008)	Ferramenta	AGG e TaskTracker.
Winkler e Pilgrim (2009)	Não consta	PRO-ART; Toor (*Traceability Object-Oriented Requirements*).
Merilinna e Yrjonen (2010)	*Framework*	SIG *Analysis Tool*
Loniewski *et al.* (2011)	Metodologia	Não utiliza.
Sanchez *et al.* (2011)	*Framework*	Não utiliza.
Siegert *et al.* (2013)	Metodologia	MDEREQTraceTool.

Uma análise comparativa dos artigos em relação às preocupações pode ser identificada no Quadro 3.4, como por exemplo, se tratam ou não das mudanças sofridas pelos requisitos ao longo do projeto, utilizam um ou mais metamodelos e quais dimensões da rastreabilidade são tratadas.

Quadro 3.4: Análise comparativa dos artigos.

	MUDANÇA DE REQUISITOS	PROPÕE METAMODELO	DIMENSÕES TRATADAS
Almeida *et al.* (2006)			
Valderas e Pelechano (2008)			
Winkler e Pilgrim (2009)	X	Outros Autores	O quê; Quem; Onde; Como; Quando; Por quê.
Merilinna e Yrjonen (2010)	X		
Loniewski *et al.* (2011)			
Sanchez *et al.* (2011)	X	Outros Autores	
Siegert *et al.* (2013)	X		

Quatro das sete propostas mencionaram uma ou mais ferramentas para apoio de rastreabilidade de requisitos, porém não apresentam como funciona uma integração entre as ferramentas listadas e a solução proposta. Um outro fator considerado preocupante é que, também, quatro delas não lidam com a alteração dos requisitos. O Gráfico 3.1 apresenta uma visão por ano e autores das dimensões tratadas.

Pode-se concluir por meio desta revisão sistemática que as propostas encontradas nem sempre fornecem uma solução bem definida para desenvolver um modelo ou tratar da rastreabilidade. Um dos fatores que contribuem para isso, é que, delas, somente uma trata das dimensões. Não foi encontrada uma classificação das informações que se deseja rastrear e que justifique as perguntas formuladas para identificar as informações a serem rastreadas. Outro fator observado é que as soluções não fornecem ou aplicam metamodelos para desenvolver

modelos de rastreamento específicos. Logo, conclui-se que as propostas apresentadas não fornecem mecanismos completos para rastreabilidade de requisitos no ambiente de MDD.

3.6 Considerações Finais do Capítulo

Este capítulo apresentou uma revisão teórica sobre a ER apontando seus artefatos e processos. Dentre as principais atividades e processos descritos, destaca-se a rastreabilidade de requisitos que pertence ao processo de Gerência de Requisitos. Isso se deve ao fato de que ela está diretamente ligada à qualidade do *software*, bem como ao acompanhamento e descrição da vida de um requisito que foi levantado, analisado, documentado e validado nas demais fases.

Gerentes e desenvolvedores podem se beneficiar com a aplicação correta da rastreabilidade. Por meio dela, é possível estimar variações em cronogramas, avaliar impactos de alterações de requisitos, auxiliar na verificação e validação, identificar a alocação de requisitos a componentes de *software,* analisar custos e prazos e gerenciar riscos e reuso de artefatos.

No capítulo também foi apresentada uma análise de metamodelos encontrados na literatura, bem como, os resultados da revisão sistemática de literatura discutida no Capítulo 1 deste trabalho. Cada um com seus próprios conceitos e relações. O objetivo geral dos metamodelos encontrados é melhorar a reutilização e servir como um modelo estruturado de referência. Entretanto, eles não abrangem as características essenciais da rastreabilidade, sobretudo, quando o foco é MDD. Dentre os modelos apresentados, alguns não tratam de MDD, enquanto que outros não mencionam as dimensões dos requisitos e elos entre eles.

O próximo capítulo apresenta o *framework* R2MDD que busca uma solução para problemas de rastreabilidade de requisitos no contexto de MDD com foco nas características importantes da rastreabilidade, como as dimensões e elos da rastreabilidade.

CAPÍTULO 4

FRAMEWORK R2MDD

Este capítulo descreve o *framework* R2MDD e sua estrutura, que possui o intuito de apoiar a rastreabilidade e o monitoramento de requisitos para o MDD, alinhado às práticas de Engenharia de Requisitos.

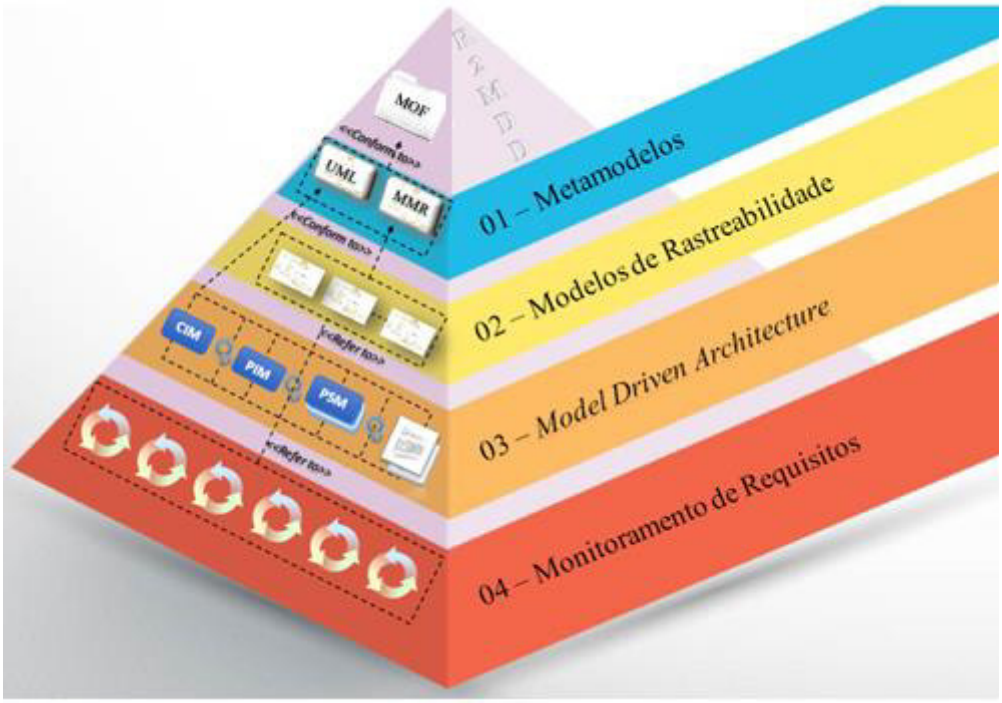
4.1 R2MDD

A escolha de um *framework* para o desenvolvimento desta pesquisa se deve pela flexibilidade, por ser considerado como uma estrutura incompleta que deixa espaço para outras práticas e ferramentas e que fornece a maior parte do processo exigido.

A fim de atingir seu objetivo, o *framework* é baseado no uso de técnicas de modelos. Diante disso, buscou-se proporcionar uma estrutura que facilite a especificação e identificação dos artefatos e as relações entre os interessados pela rastreabilidade. Além disso, optou-se pela criação de um MMR que desempenha um papel importante no *framework.* Este deverá armazenar as informações necessárias para a rastreabilidade dos requisitos no processo de MDD, bem como poderá ser considerado um repositório de dados definido por meio de XML.

A definição de MMR ou de qualquer outra estrutura para armazenar informações de rastreamento exige a identificação de quais produtos (casos de uso, características, componentes ou interfaces) são necessários rastrear, bem como informações de relacionamentos entre esses artefatos (como dependências ou influências). No entanto, essas informações podem variar de projeto para projeto a depender de sua natureza. O esquema geral do R2MDD é ilustrado na Figura 4.1.

Figura 4.1: ***Framework*** **R2MDD.**



O R2MDD está organizado em quatro níveis: 01 – Metamodelos; 02 –Modelos de Rastreabilidade; 03 – *Model Driven Architecture*; e, 04 – Monitoramento de Requisitos. A representação em pirâmente do R2MDD ocorre para demonstrar uma hierarquia entre os níveis. No topo do R2MDD encontra-se o *Meta-Object Facility*, definindo os elementos básicos para a criação de metamodelos.

No Nível 01, há a definição de um metamodelo de rastreabilidade (MMR) que serve de suporte para os demais níveis. O MMR define uma gramática de apoio para a geração dos modelos de rastreabilidade durante as transformações e será apresentado na seção 4.2.1 deste trabalho. Além do metamodelo de rastreabilidade, o Nível 01 armazena o metamodelo UML, que está em conformidade com o meta-metamodelo MOF.

O Nível 02 é responsável por armazenar as informações de rastreabilidade de todos os eventos ocorridos durante o Nível 03. Nesse nível, são gerados modelos que evoluem a cada transformação e em cada mudança no estado dos requisitos. Todos os modelos gerados devem estar em conformidade com o MMR. Além disso, eles se referem aos modelos gerados no Nível 03.

O primeiro modelo deve ser criado manualmente pelo utilizador e, a partir disso, todos os *links* podem ser criados de forma automática como parte da transformação do modelo da fonte ao destino. As transformações dos modelos gerados para os modelos de

rastreabilidade podem ocorrer por meio de regras de transformação definidas por um Engenheiro de *Software*.

Ao mesmo tempo em que são feitas as criações e transformações de modelos mais próximos à plataforma, deve ser armazenado um registro das ligações entre os elementos envolvidos em algum repositório de rastreabilidade pré-definido na fase inicial do processo de desenvolvimento.

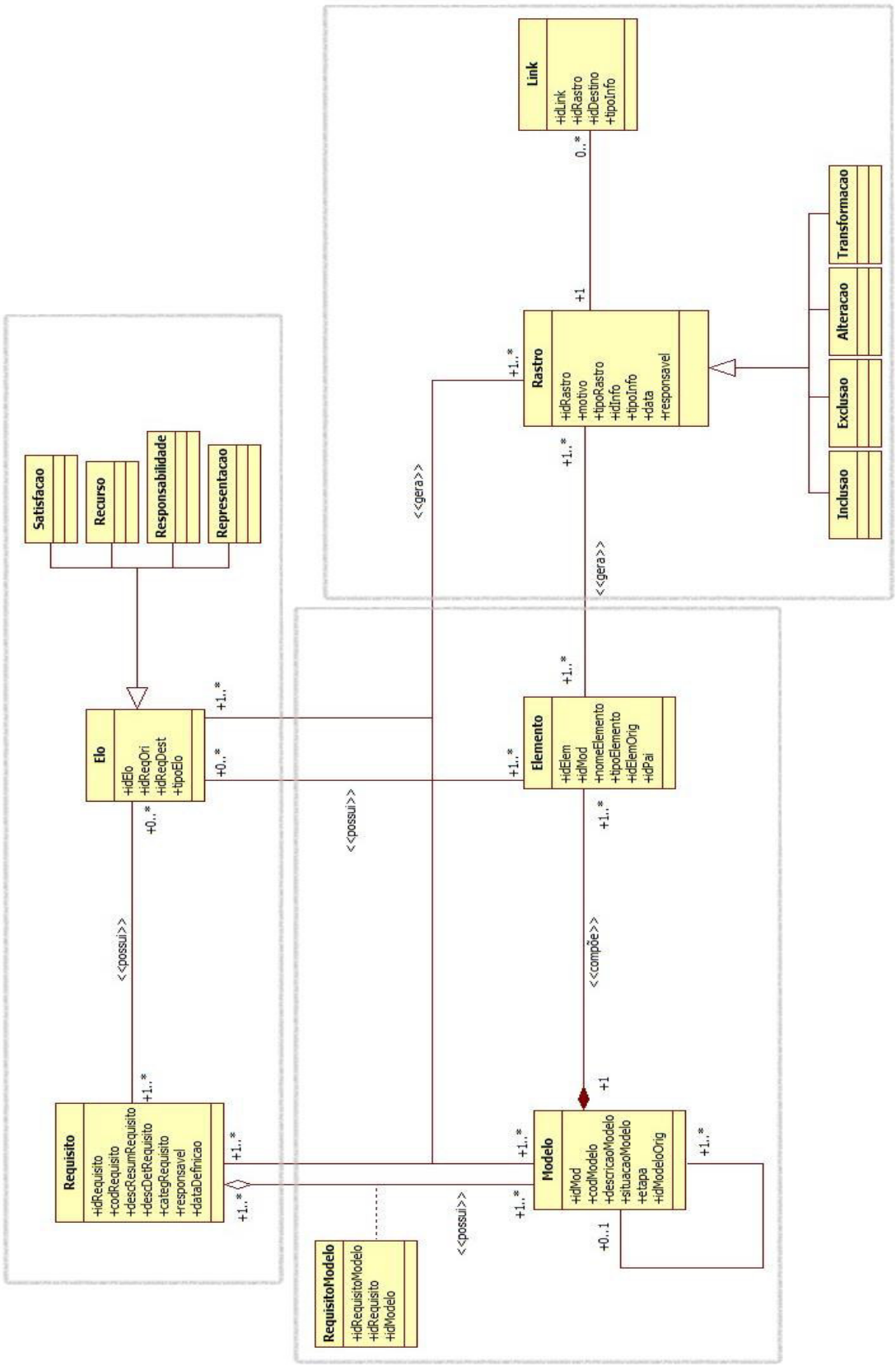
O Nível 03 está relacionado à etapa de geração e transformação de modelos, seguindo a arquitetura MDA. Neste nível os requisitos levantados são as entradas e o produto de *software,* a saída. Os modelos de rastreabilidade do Nível 02 são gerados de forma paralela à geração dos modelos do Nível 03. É importante destacar que eles devem estar em conformidade com o metamodelo UML do Nível 01, o qual deve estar em conformidade com o meta-metamodelo do topo do R2MDD.

O Nível 04 prevê o monitoramento dos requisitos através do acompanhamento dos modelos gerados. Em qualquer momento do Nível 03 o estado de cada requisito pode ser identificado. Nesse nível é possível identificar quais foram os componentes alocados aos requisitos; os artefatos envolvidos no desenvolvimento; determinar as implicações que uma mudança pode acarretar; avaliar impactos e quais requisitos serão afetados em caso de alguma modificação; bem como prever custos; tempo de modificação; gerar matrizes de rastreabilidade; certificar a validade dos requisitos; e, e conferir se a implementação resultante é compatível com os requisitos de entrada. Essas informações são obtidas a partir dos modelos gerados no Nível 02.

4.2 Metamodelo de Rastreabilidade

O MMR provê uma gramática de apoio à geração dos modelos de rastreabilidade originados no Nível 03 do *framework*. Para definir o metamodelo, foram utilizados os conceitos de rastreabilidade e de MDD abordados neste trabalho, que também está em conformidade com o meta-metamodelo UML. O metamodelo de rastreabilidade proposto está representado na Figura 4.2.

Figura 4.2: Metamodelo do R2MDD.

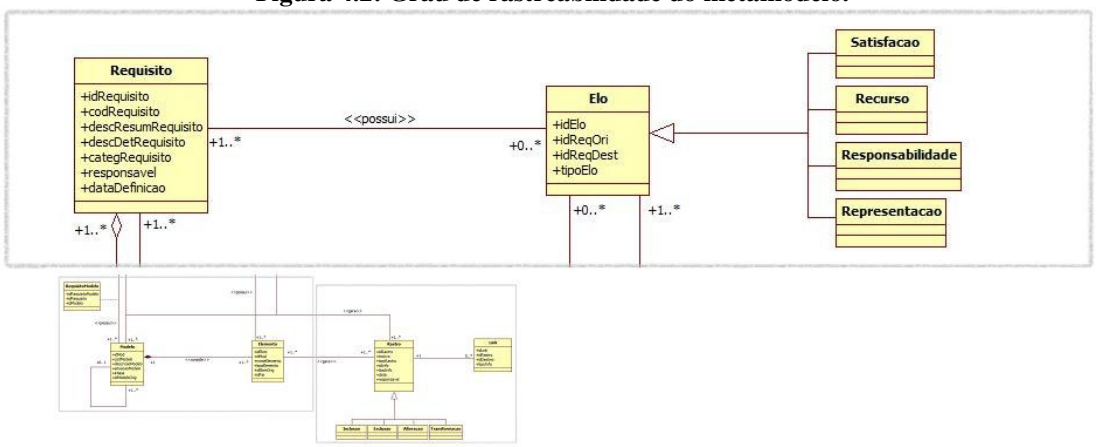


O MMR busca relacionar cada requisito aos eventos ocorridos em cada transformação, gerando os traços de rastreabilidade. Além disso, ele também apresenta o grau de rastreabilidade entre cada transformação, que é representado através dos tipos de ligação.

4.2.1 Elementos do MMR

O suporte à rastreabilidade emprega mesma classificação utilizada por Toranzo (2002) através dos seguintes tipos de ligação: satisfação, recurso, responsabilidade e representação. Na Figura 4.2 o grau de rastreabilidade por meio dos tipos de ligações e a definição dos requisitos do modelo proposto são apresentados.

Figura 4.2: Grau de rastreabilidade do metamodelo.



Os tipos de ligação especificam um elo entre requisitos e elementos. A partir dos elos é possível identificar quais requisitos serão afetados com uma mudança e quais são as ligações entre eles. No metamodelo, um ou mais requisitos estão associados a um ou mais elos entre requisitos, os quais devem satisfazer um dos tipos de ligações de rastreabilidade estabelecidos pelo metamodelo.

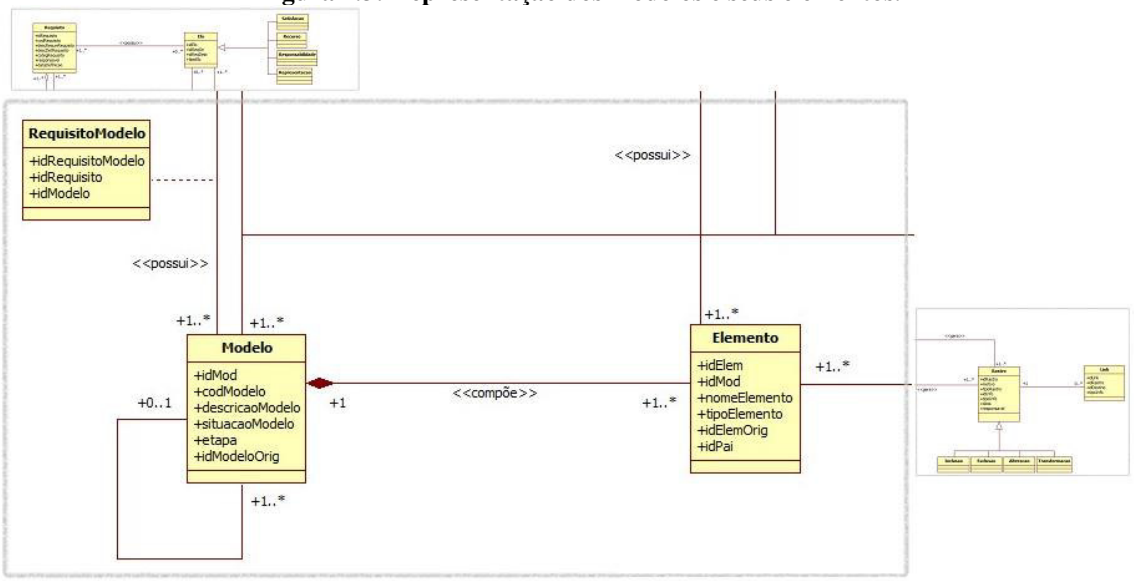
A metaclasse "Requisito" armazena as informações levantadas no início do projeto junto aos *stakeholders* e analistas. Ela representa as necessidades do produto de *software* a ser gerado. Cada requisito deve possuir uma categoria, descrição, quem é o responsável e a data de definição. Requisitos relacionam-se com outros requisitos e/ou elementos através de elos, que podem ser satisfação, recurso, responsabilidade e representação. Um ou mais requisitos deve estar associado a um ou mais modelos que serão transformados durante a MDA. Esses modelos são compostos por elementos que podem se relacionar entre si.

Os elementos da Figura 4.2 do metamodelo que representam o grau de rastreabilidade são:

- Requisito: sumarização de uma interpretação da comunicação do usuário com o analista, ou seja, a representação das necessidade do usuário. Seus atributos são:
  - codRequisito: código do requisito do tipo texto que segue a seguinte regra: em caso de Requisito Funcional, o código gerado será composto pelas siglas RF seguido de um número sequencial que irá de 001 à 999, ou seja, estará entre RF001 e RF999, caso o requisito seja Não Funcional, a sigla utilizada será RN, seguido do número sequencial.
  - descResumRequisito: compreende a descrição resumida do requisito em formato de texto;
  - descDetRequisito: texto referente à descrição geral do requisito sob o ponto de vista do *stakeholder*;
  - categRequisito: categoria do requisito, podendo ser Requisito Funcional (RF) e Requisito Não-Funcional (RFN), por exemplo.
  - Responsavel: informação relacionada a quem definiu o requisito; e,
  - DataCriacao: data em que foi definido do tipo data e hora.
- Elo: representa o tipo de ligação entre os requisitos e elementos. Um elo pode ser do tipo: satisfação, recurso, responsabilidade ou representação.

Além do grau de rastreabilidade e da definição dos requisitos, o metamodelo abrange informações sobre os modelos que serão transformados e seus elementos. Essas informações podem ser visualizadas na Figura 4.3.

Figura 4.3: Representação dos modelos e seus elementos.

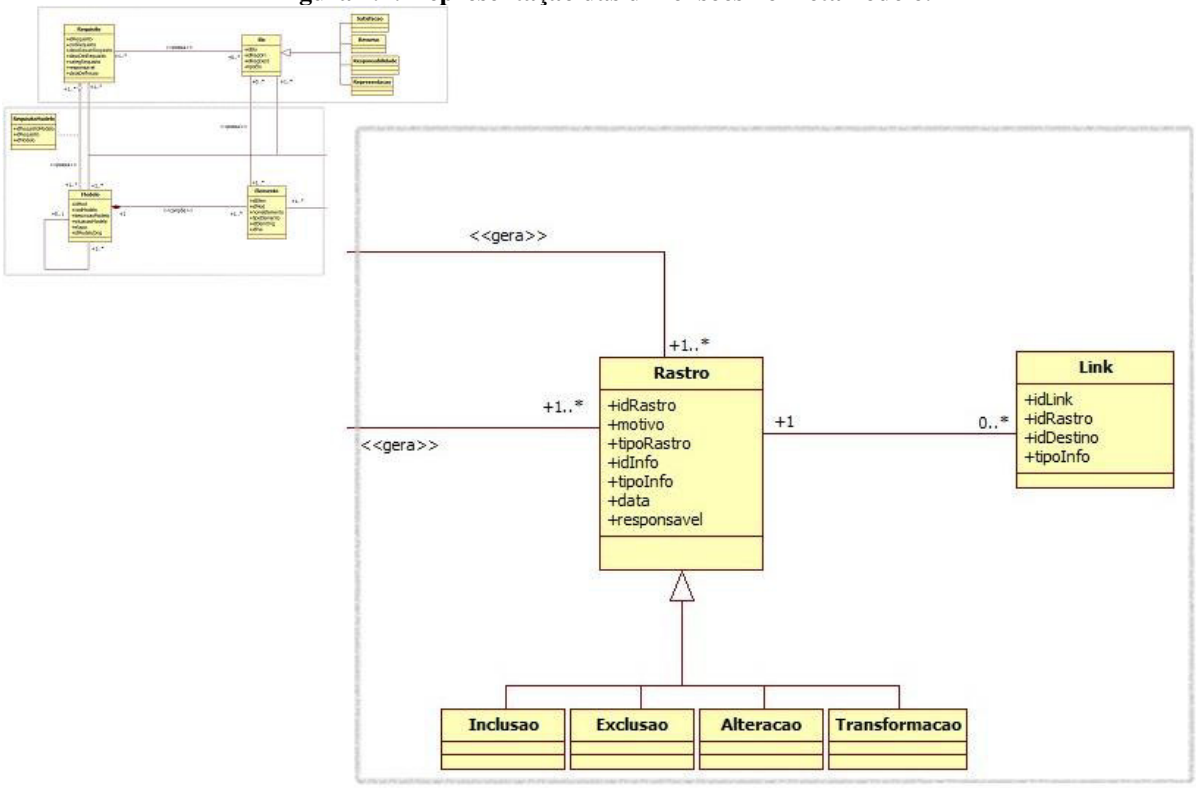


Os elementos da Figura 4.3 podem ser definidos da seguinte forma:

- Modelo: informações dos modelos que estão sendo transformados. Cada modelo pode estar associado a um ou mais requisitos. Na metaclasse modelo, são importantes as informações de descrição e situação.
- RequisitoModelo: classe associativa que realiza a ligação entre os modelos e os requisitos.
- Elemento: qualquer informação que faz parte do modelo. Um elemento pode ser uma classe, um trecho de código, entre outras informações.

As cinco dimensões discutidas no Capítulo 3 deste trabalho, bem como os rastros gerados e suas informações também estão representadas no metamodelo proposto. A Figura 4.4 representa a parte do metamodelo que trata dessas dimensões.

Figura 4.4: Representação das dimensões no metamodelo.



Essa é considerada a parte principal do metamodelo, uma vez que é responsável por manter o histórico de todas as mudanças ocorridas, sejam elas nos requisitos, nos elos ou nos elementos que estão sendo modificados. Os elementos apresentados na Figura 4.5 são:

- Rastro: resultado de um evento gerado ou ocorrência, registro de elementos e seus estados. Um rastro deve responder as seis questões tratadas pela rastreabilidade.
  - O Que? Pode ser respondido pelos campos: tipoRastro e tipoInfo. O campo tipoRastro pode ser uma Inclusão, Exclusão ou Alteração. E o campo tipoInfo pode ser um elemento do modelo, um requisito ou um elo.
  - Quem? É respondido por meio do responsável da metaclasse InfRastro;
  - Como? Essa pergunta tem como resposta o campo tipoInfo, que pode ser Inclusão, Exclusão ou Alteração de artefato;
  - Onde? Os campos IdInfo da metaclasse Rastro e idDestino da metaclasse Link respondem essa pergunta. Eles armazenam informações do elemento que foi afetado origem e do destino;

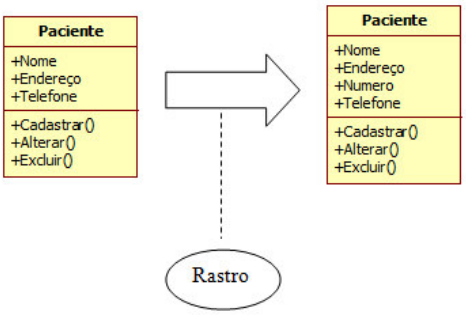
- Porque? O campo motivo representa o motivo que gerou o rastro, e, por isso, responde a esta pergunta;
- Quando? Respondida por meio do campo data, de InfRastro.

- Link: metaclasse que faz uma ligação do elemento que originou o rastro e o elemento no qual ele se transformou, bem como seu tipo.

A metaclasse "Rastro" representa o evento ocorrido no modelo que está sendo transformado ou no requisito que está sendo alterado. Os eventos são acontecimentos sobre determinado requisito ou elemento, sendo eles de influência interna ou externa. Cada rastro é gerado sempre que houver mudança no estado de um requisito, de forma conceitual ou de elementos que fazem parte do modelo associado a esse requisito, inclusive durante as transformações no Nível 03. A alteração em um requisito, elo, modelo ou elemento de modelo gera rastros, bem como as transformações.

Um exemplo de geração de rastro é a inclusão de um novo atributo em uma determinada classe (Figura 4.5), ou de uma nova classe em um modelo. A alteração do nome de um atributo também deve gerar rastro, o qual deve armazenar o estado anterior e atual do elemento alterado.

Figura 4.5: Exemplo de representação de um rastro.



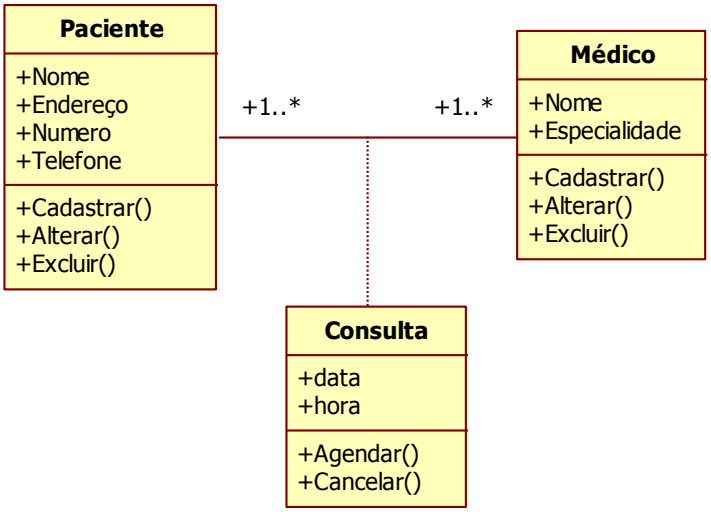
A Figura 4.5 apresenta o estado anterior do elemento "Paciente" e seu estado atual, com a inclusão de um novo atributo "Numero". Essa alteração gerou um rastro de inclusão de atributo.

4.3 Monitoramento de Requisitos

O Nível 04 trata do monitoramento e acompanhamento dos requisitos durante a geração ou transformações de modelos. O monitoramento ocorre a partir dos eventos registrados nos modelos do Nível 02.

Com os modelos gerados, diversos tipos de dados podem ser obtidos, por exemplo: matriz de rastreabilidade, histórico do requisito, elementos dos requisitos por etapa da MDA, previsão de impactos, novos diagramas de classes, entre outros. Além disso, podem ser gerados relatórios com diferentes *templates* e informações, atendendo à necessidade de cada *stakeholder*. Para exemplificar o monitoramento de requisitos, a Figura 4.6 ilustra um exemplo de diagrama de classes que sofrerá alterações.

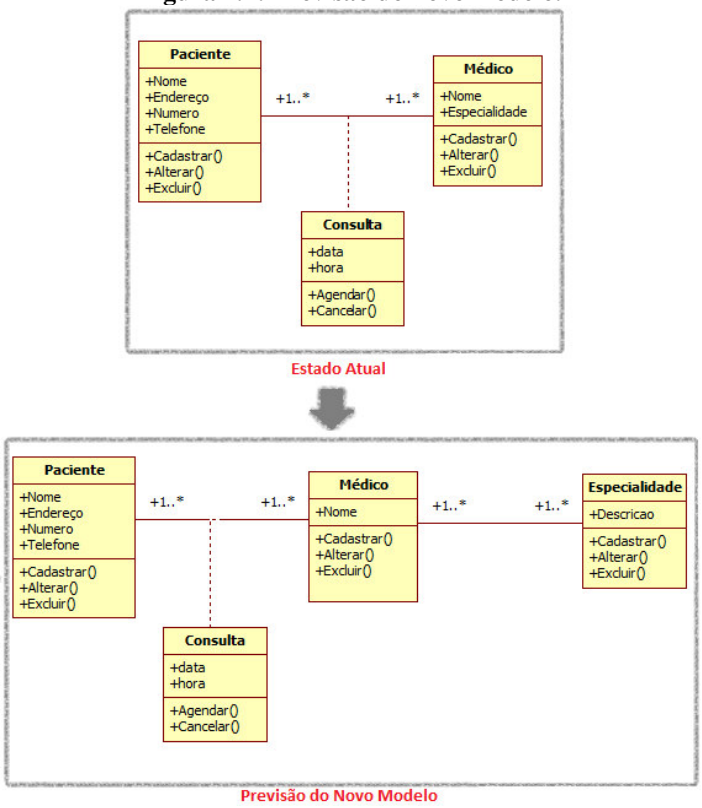
Figura 4.6: Exemplo de diagrama de classes.



O diagrama representa o modelo M001, referente ao requisito RF001, responsável pela marcação de consultas em um ambiente hospitalar. Para atendê-lo, é necessário o cadastro de pacientes e médicos. Um ou mais pacientes podem ser consultados por um ou mais médicos e cada médico possui uma especialidade.

No caso estudado, no entanto, surgiu a necessidade de alterar o requisito para que um médico possua uma ou mais especialidades. Sendo assim, deverá ser incluída uma classe "Especialidade" que terá uma associação de um para muitos com a classe "Médico". A Figura 4.7 apresenta uma previsão do modelo resultante da aplicação dessas alterações.

Figura 4.7: Previsão do novo modelo.



Nota-se que houve a remoção do atributo "especialidade" da classe "Médico", enquanto que foi incluída uma classe denomidada "Especialidade" e uma associação entre essa nova classe e "Médico". Levando em consideração que apenas a etapa CIM foi executada, o possível relatório de impacto de alteração de artefatos pode ser identificado na Figura 4.8.

Figura 4.8: Relatório de Impacto de Modificação de Artefatos.

Relatório de Impacto de Alteração de Elementos		
Requisito:	RF001	
Elemento:	especialidade	
Tipo Alteração:	Exclusão	
Motivo:	Criação de uma nova classe	
Data:	01/05/2016	
Elementos que serão afetados		
Etapa	**Local Afetado**	**Nome**
CIM	Requisito	RF001
	Modelo	M001
	Elemento - Atributo	especialidade
	Elemento - Classe	Médico
Total de Elementos Afetados:	4	
Percentual de Impacto:	17%	

No *layout* proposto, o termo "Artefato" pode ser qualquer elemento, modelo, requisito ou elo modificado/afetado. Por meio dele, é possível identificar os impactos decorrentes de alterações em artefatos relacionados ao requisito em cada etapa da MDA. Da mesma maneira, são apresentados a quantidade de artefatos afetados e o percentual de impacto, calculado com base na divisão da quantidade de artefatos afetados pelo total de artefatos ligados ao requisito. Todas as informações são obtidas através de relações entre os modelos de rastreabilidade gerados durante as transformações.

Com os modelos gerados, também é possível montar uma matriz de rastreabilidade de requisitos. Um possível protótipo pode ser representado na Figura 4.9.

Figura 4.9: Matriz de Rastreabilidade.

A matriz apresentada na Figura 4.9 serve para visualizar os relacionamentos entre os requisitos e apresentar seus elos. Outras matrizes também podem ser geradas, como por exemplo, a matriz rastreabilidade de artefatos ligados ao requisito.

4.4 Considerações Finais do Capítulo

Este capítulo apresentou o R2MDD e sua estrutura, composta por quatro níveis: metamodelo de rastreabilidade de requisitos, modelos de rastreabilidade, MDA e monitoramento de requisitos.

O R2MDD tem como ponto forte a capacidade de rastrear requisitos durante a execução das transformações de modelos. Além disso, foi proposto um metamodelo de apoio ao R2MDD que trata dos principais conceitos de rastreabilidade. Por meio dele é possível identificar respostas às dimensões da rastreabilidade e o estado dos elementos associados ao requisito.

Algumas limitações foram encontradas, como, por exemplo, a pouca quantidade de artigos e trabalhos que tratam do tema, bem como de ferramentas de apoio para o desenvolvimento desta pesquisa.

Para avaliar a validade do R2MDD, fez-se necessário um experimento com informações de um ambiente real. O próximo capítulo apresenta o experimento deste trabalho, aplicado no Sistema de Triagem NeoNatal do Hospital Universitário de Sergipe (HU-UFS), buscando provar a hipótese de que é possível rastrear os requisitos durante os processos do MDD.

CAPÍTULO 5

CASO EXEMPLO UTILIZANDO O R2MDD

Este capítulo descreve um experimento realizado seguindo o R2MDD com o objetivo de validar a aplicabilidade do *framework.* O objeto de estudo foi o Serviço de Referência de Triagem Neonatal (STRN) do Hospital Universitário da Universidade Federal de Sergipe (HU - UFS). Para realizar o estudo, foi utilizado o modelo de processo *Qualitas*, proposto por Almeida (2014), que será descrito neste capítulo.

5.1 O Hospital Universitário e o Programa de Triagem Neonatal (PNTN)

O HU-UFS está vinculado à Universidade Federal de Sergipe e foi criado para prestar "assistência médico-hospitalar e contribuir com o desenvolvimento das atividades de natureza preventiva e extensiva" (HU-UFS, 2012) no estado. Segundo (HU-UFS, 2012), o hospital serve de alicerce para atividades acadêmicas de diversos cursos da UFS (áreas médica e multiprofissional), atua na participação e colaboração de programas nacionais de saúde e promove parcerias com órgãos públicos (das esferas federal, municipal, estadual).

O hospital não realiza atendimentos particulares ou através de planos de saúde. Na verdade, o HU-UFS presta serviços para o Sistema Único de Saúde (SUS) sendo referência em atendimentos de média e alta complexidade no ambulatório e em relação a exames complementares, diagnóstico e internação.

O Programa de Triagem Neonatal (PNTN) da Portaria GM/MS 822/2001 possui o objetivo de realizar exames de diagnósticos para detecção precoce de doenças no período neonatal (entre 0 e 28 dias de vida) e o acompanhamento e tratamento de pacientes durante toda a vida (ALMEIDA, 2014).

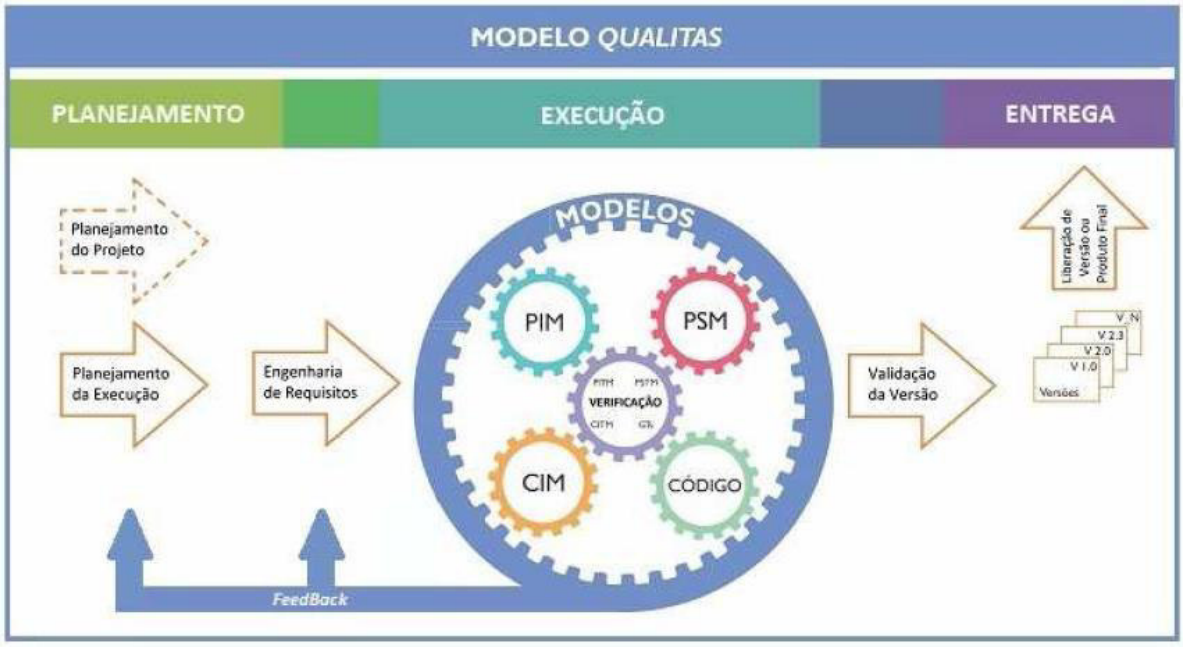
Segundo o Ministério da Saúde (2002), os exames são obrigatórios em todos os recém-nascidos (RN) vivos. Eles se destacam por buscar a ampliação da gama de patologias triadas, como fenilcetonúria, hipotireoidismo congênito, anemia falciforme e outras hemoglobinopatias e fibrose cística. A meta principal do PNTN é a diminuição da morbimortalidade causada pelas patologias triadas.

Em 5 de Junho de 2013 o HU-UFS ampliou os serviços prestados a toda rede pública do estado passando a ser habilitado como Serviço de Referência em Triagem Neonatal (SRTN) em Sergipe através da Portaria Nº 1.082, que atua na fase de detecção de fibrose cística do PNTN (ALMEIDA, 2014).

5.2 O Modelo de Processo *Qualitas*

O *Qualitas* é um processo de desenvolvimento de sistemas de *software* iterativo e incremental proposto por Almeida (2014) com o intuito orientar pequenos e médios projetos de *software* orientado a modelos (Figura 5.1).

Figura 5.1: O Modelo de Processo *Qualitas*.



Fonte: Almeida (2014).

O ciclo de vida do Qualitas está dividido em três fases: Planejamento, Execução e Entrega. As etapas da fase de execução são: Engenharia de Requisitos, CIM, CITM, PIM, PITM, PSM, PSTM, código, geração de testes e validação de versão. Sendo que o CITM consiste na geração de modelo de testes para o CIM; o PITM na geração de modelos para a execução dos testes do PIM e o PSTM na geração de modelos de teste para cada PSM. O Quadro 5.1 apresenta um resumo das atividades do modelo *Qualitas*.

Quadro 5.1: Resumo das atividades do modelo *Qualitas*.

Fases	Etapas	Atividades
Fase Planejamento	Planejamento do Projeto	Construir o Plano de Gestão do Projeto.
	Planejamento da Execução	Produzir o Plano de Execução.
Execução	Engenharia de Requisitos	Produzir o Documento de Requisitos.
	CIM	Construir os Modelos Independentes de Computação.
	CITM	Gerar os Modelos de Testes Independentes de Computação.
		Testar os Modelos Independentes de Computação.
		Analisar os Resultados dos Testes.
	PIM	Gerar o Modelo Independente de Plataforma.
	PITM	Gerar os Modelos de Testes Independentes de Plataforma.
		Testar os Modelos Independentes de Plataforma.
		Analisar os Resultados dos Testes.
	PSM	Gerar os Modelos Específicos de Plataforma.
	PSTM	Gerar os Modelos de Testes Específicos de Plataforma.
		Testar os Modelos Específicos de Plataforma.
		Analisar os Resultados dos Testes.
	Geração de Código	Gerar o Código Fonte a partir do PSM.
		Complementar, se necessário, o Código Fonte.
	Geração de Testes	Gerar os Testes a partir do código.
	Validação da Versão	Validar se a versão está atendendo ao que foi especificado.
		Avaliar os Resultados da Execução dos Testes.
Fase Entrega	Liberação de Versão ou Produto Final	Disponibilizar cada versão validada, ou ao término do projeto, o produto final para o cliente.

Fonte: Almeida (2014).

Levando em consideração que as ferramentas existentes, em sua maioria, não suportam todas as transformações de forma automática, o processo sugere a utilização de trabalho manual durante transformações a depender da ferramenta adotada. Nesse caso, é importante compreender que o esforço e o prazo do projeto deverão ser maiores, fator que deve ser destacado na fase de planejamento.

5.3 O Caso Exemplo

O objeto de estudo desta pesquisa é o PNTN do HU-UFS. Para realizar o estudo, foi escolhido um requisito da lista de requisitos do Quadro 5.2 (ALMEIDA, 2014).

Quadro 5.2: Requisitos PNTN.

ID	Requisitos Funcionais
RF001	O sistema deverá possuir cadastro de usuários, perfis de usuários e os diferentes acessos que estes possuem.
RF002	O sistema deverá possuir cadastro de pacientes e de responsáveis a partir dos cartões recebidos.
RF003	O sistema deverá possuir cadastro de unidades de saúde.
RF004	O sistema deverá permitir o cadastro dos cartões dos recém-nascidos.

RF005	O sistema deverá manter os registros históricos de cada paciente, principalmente os reconvocados e os casos positivos confirmados.
RF006	O sistema deverá possuir a identificação unívoca de cada amostra recebida, assim como a data de recebimento e origem, permitindo sua rastreabilidade.
RF007	O sistema deverá possuir o encaminhamento ordenado das amostras ao laboratório, de forma a manter relação com a remessa e a identificação original.
RF008	O sistema deverá permitir o registro de forma segura e unívoca os resultados dos testes de cada amostra, registrando através de senhas, o responsável técnico pela liberação dos mesmos.
RF009	O sistema deverá permitir identificar automaticamente os casos que deverão ser reconvocados.
RF010	O sistema deverá disponibilizar automaticamente os resultados dos exames realizados, de forma a evitar erros de transcrição.
RF011	O sistema deverá disponibilizar rapidamente os resultados, no máximo em sete dias após o recebimento da amostra, remetendo-os à rede de coleta de forma segura e auditável.
RF012	O sistema deverá manter mecanismos de controle do retorno dos casos reconvocados até o diagnóstico final.
RF013	O sistema deverá manter atualizados os cadastros de casos positivos para cada uma das patologias detectadas.
RF014	O sistema deverá permitir a identificação unívoca de cada caso positivo confirmado, permitindo sua rastreabilidade.
RF015	O sistema deverá permitir o encaminhamento ordenado da confirmação diagnóstica ao laboratório, de forma a manter relação com a identificação original.
RF016	O sistema deverá permitir registrar de forma segura e unívoca as informações contidas no prontuário de cada paciente, registrando através de senhas, o responsável técnico pela informação.
RF017	O sistema deverá permitir identificar automaticamente os casos que deverão ser convocados para nova consulta de acompanhamento.
RF018	O sistema deverá manter mecanismos de controle do retorno dos pacientes às consultas agendadas.
RF019	O sistema deverá manter atualizados os cadastros de casos positivos para cada uma das patologias detectadas.

Fonte: Almeida (2002).

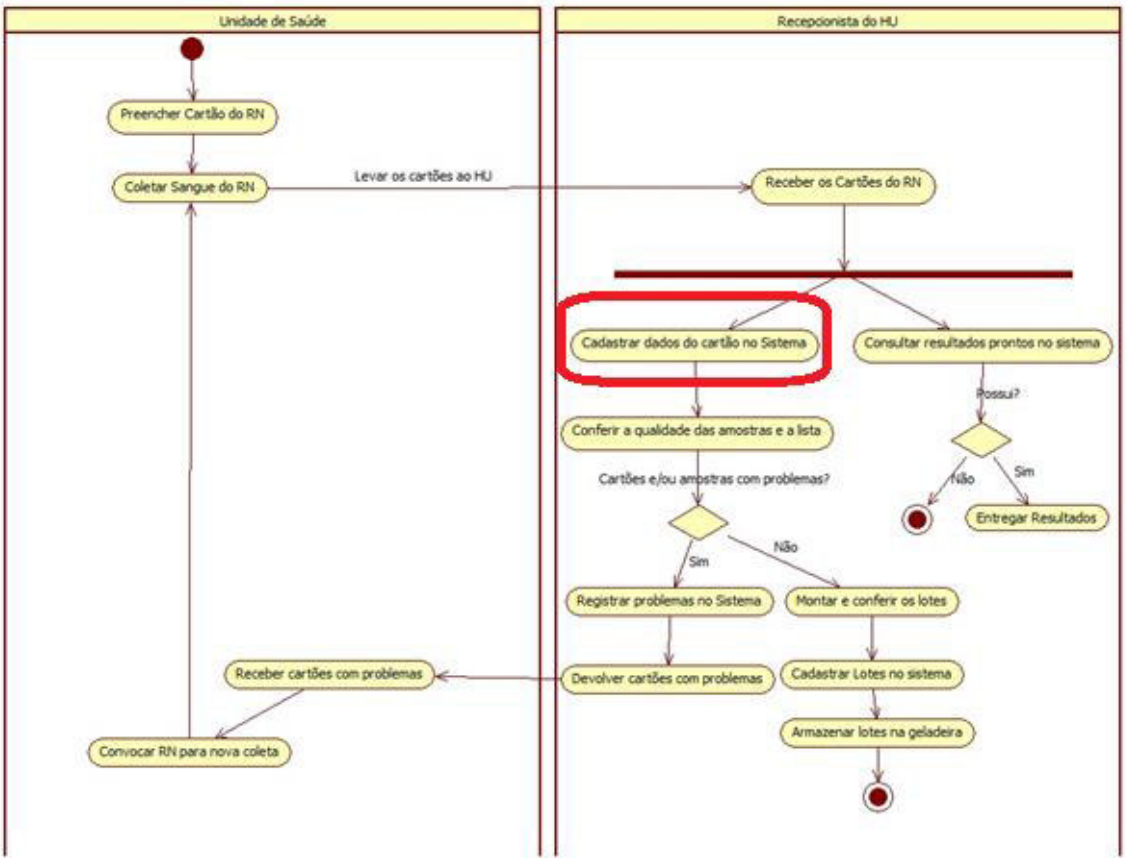
Para validar o R2MDD, foi utilizado o requisito funcional RF004, levantado durante a validação das etapas de planejamento do projeto, planejamento da execução e engenharia de requisitos do modelo *Qualitas* por Almeida (2014). A aplicação do *framework* poderá ser validada nas próximas etapas do modelo.

5.3.1 Etapa CIM

Durante a etapa CIM, Almeida (2014) fez a modelagem *To Be* dos seguintes processos: Colher sangue do recém-nascido e encaminhar para realização do exame, realizar exames TSH e PKU, e, realizar exame de hemoglobina. As etapas de colher sangue do recém

nascido e encaminhar para realização dos exames podem ser representadas por meio do CIM modelado em UML, para isso a autora criou um diagrama de atividades (Figura 5.2).

Figura 5.2: Colher sangue no RN e encaminhar para a realização do exame.



Fonte: (ALMEIDA, 2014).

A atividade “Cadastrar dados do cartão do sistema” está relacionada ao requisito que será utilizado para o caso exemplo desta pesquisa. Neste momento, é especificado o diagrama de classes para o modelo CIM (Figura 5.3). Essa atividade consiste em gerar cadastros com informações do RN no sistema. Sendo assim, as informações necessárias são: identificação do RN, registro no STRN, tipo de cartão, posto, data de coleta, lote, antibiótico, se o RN é prematuro, se é necessária transfusão e quem é o responsável pela coleta. Com essas informações, e seguindo o R2MDD, o primeiro modelo de rastreabilidade do CIM pode ser gerado.

Figura 5.3: Modelo CIM referente ao requisito "Cadastrar dados do cartão no sistema".

Inicialmente, é preciso identificar possíveis informações para o modelo de rastreamento da etapa CIM do requisito "Cadastrar dados do cartão no sistema". No modelo, considerou-se que o requisito RF004 possui um elo de satisfação com o RF002, e por isso, esse requisito foi registrado. Com base nas informações levantadas, o modelo de rastreabilidade resultante do CIM é representado nos quadros 5.3, 5.4 e 5.5.

Quadro 5.3: Modelo de Rastreabilidade CIM.

Metaelemento do R2MDD	Atributo do Metaelemento do R2MDD	Elemento do Modelo de Rastreabilidade gerado no CIM
Requisito	idRequisito	1
	codRequisito	RF002
	descResumRequisito	Cadastrar pacientes e responsáveis a partir de cartões recebidos.
	descDetRequisito	O sistema deverá possuir cadastro de pacientes e de responsáveis a partir de cartões recebidos.
	categRequisito	Requisito Funcional
	Responsável	Carla
	dataDefinicao	17/04/16
Requisito	idRequisito	2
	codRequisito	RF004
	descResumRequisito	Cadastrar dados do cartão no sistema.
	descDetRequisito	O sistema deverá permitir o cadastro dos cartões dos recém-nascidos.
	categRequisito	Requisito Funcional
	Responsável	Carla
	dataDefinicao	17/04/16
Elo	IdElo	1
	idReqOrig	1
	idReqDest	2
	tipoElo	Satisfação
Modelo	IdMod	1
	codModelo	M001
	descricaoModelo	Primeiro modelo de rastreabilidade gerado no CIM.
	situacaoModelo	Ativo
	Etapa	CIM
	idModOrig	*Null*
RequisitoModelo	idRequisitoModelo	1
	idRequisito	2
	idModelo	1

Os elementos do modelo de rastreabilidade gerados no CIM em conformidade com a metaclasse "Elemento" são apresentados no Quadro 5.4.

Quadro 5.4: Modelo de Rastreabilidade CIM – Elementos.

Elemento						
Atributo do Metaelemento do R2MDD	**IdElem**	**IdMod**	**nomeElemento**	**tipoElemento**	**idElemOrig**	**idPai**
Elemento do Modelo de Rastreabilidade gerado no CIM	1	1	CartaoRN()	Classe	*null*	
	2	1	recemNascido	Atributo	*null*	1
	3	1	regSTRN	Atributo	*null*	1
	4	1	tipoCartao	Atributo	*null*	1
	5	1	postoColeta	Atributo	*null*	1
	6	1	dataColeta	Atributo	*null*	1
	7	1	lote	Atributo	*null*	1
	8	1	antibiotico	Atributo	*null*	1
	9	1	prematuro	Atributo	*null*	1
	10	1	transfusao	Atributo	*null*	1
	11	1	responsavelColheita	Atributo	*null*	1
	12	1	IncluirCartaoRN()	Funcao	*null*	1
	13	1	AlterarCartaoRN()	Funcao	*null*	1

Por fim, o Quadro 5.5 apresenta as informações dos rastros gerados na etapa CIM, em conformidade com a metaclasse "Rastro".

Quadro 5.5: Modelo de Rastreabilidade CIM – Rastros.

Rastros							
Atributo do Metaelemento do R2MDD	**idRastro**	**Motivo**	**tipoRastro**	**idInfo**	**tipoInfo**	**data**	**responsavel**
Elemento do Modelo de Rastreabilidade gerado no CIM	1	Incluir informações de novo requisito responsável por cadastrar dados do cartão do RN no sistema	Inclusão	2	Requisito	17/04/16	Izabella
	2	Gerar modelo M001 de rastreabilidade, na etapa CIM que atende ao requisito de cód. RF001	Inclusão	1	Modelo	17/04/16	Izabella
	3	Associar de Modelo M001 a requisito REQ001	Inclusão	1	Requisito Modelo	17/04/16	Izabella
	4	Gerar classe que irá armazenar os dados do cartão do RN	Inclusão	1	Elemento: Classe	17/04/16	Izabella
	5	Armazenar recém-nascido com inf. do paciente	Inclusão	2	Elemento: Atributo	17/04/16	Izabella
	6	Armazenar registro do RN no STRN	Inclusão	3	Elemento: Atributo	17/04/16	Izabella
	7	Armazenar tipo do cartão do RN	Inclusão	4	Elemento: Atributo	17/04/16	Izabella
	8	Armazenar sobre o posto responsável	Inclusão	5	Elemento: Atributo	17/04/16	Izabella
	9	Armazenar a data de colheita	Inclusão	6	Elemento: Atributo	17/04/16	Izabella
	10	Guardar informações de lote	Inclusão	7	Elemento: Atributo	17/04/16	Izabella

	11	Indicar o antibiotico utilizado no RN	Inclusão	8	Elemento: Atributo	17/04/16	Izabella
	12	Indicar se paciente é prematuro	Inclusão	9	Elemento: Atributo	17/04/16	Izabella
	13	Indicar se é necessária transfusão	Inclusão	10	Elemento: Atributo	17/04/16	Izabella
	14	Indicar o responsável pela colheita de sangue	Inclusão	11	Elemento: Atributo	17/04/16	Izabella
	15	Incluir informações na classe CartaoRN	Inclusão	12	Elemento: Função	17/04/16	Izabella
	16	Alterar informações na classe CartaoRN	Inclusão	13	Elemento: Função	17/04/16	Izabella

Com o modelo de rastreabilidade resultante da etapa CIM, é possível identificar informações importantes sobre rastros gerados e responder a todas as dimensões da rastreabilidade, por meio de um único rastro. As informações que correspondem ao rastro gerado para a inclusão de um modelo, por exemplo, são:

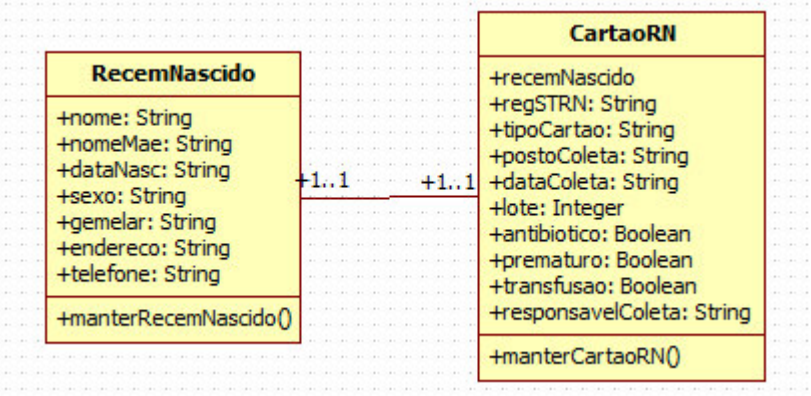
- O Que? Um rastro foi gerado após a inclusão de um modelo (idRastro = 2);
- Quem? O "responsável" foi "izabella";
- Como? Por meio da "Inclusão" de um novo "Modelo";
- Onde? Na etapa CIM, onde foram incluídas informações do requisito, modelo e elementos associados ao modelo;
- Porque? Devido à "Inclusão do modelo M001, na etapa CIM que atende ao requisito de cód. RF001";
- Quando? Em 17/04/16.

É possível, ainda, saber quais requisitos estão associados ao modelo que foi criado, bem como, em caso de uma possível necessidade de alteração de requisito, saber o que será afetado, uma vez que um "Modelo" possui relação direta com os requisitos e, também, com seus elementos.

5.3.2 Etapa PIM

Na transformação do modelo CIM para PIM, foram executadas regras de transformação definidas por Almeida (2014) que geraram o modelo PIM representado na Figura 5.5.

Figura 5.4: Modelo PIM a partir do CIM.



Fonte: (ALMEIDA, 2014).

Nessa etapa, a atividade "Cadastrar dados do cartão no sistema" resulta em duas classes na etapa PIM: "RecemNascido" e "CartaoRN". Os Quadros 5.6, 5.7 e 5.8 apresentam o resultado do modelo de rastreabilidade gerado para essa etapa.

Quadro 5.6: Modelo de Rastreabilidade PIM.

Metaelemento do R2MDD	Atributo do Metaelemento do R2MDD	Elemento do Modelo de Rastreabilidade gerado no PIM
Modelo	idMod	2
	codModelo	M002
	descricaoModelo	Primeiro modelo de rastreabilidade gerado no PIM.
	situacaoModelo	Ativo
	Etapa	PIM
	idModOrig	1

No Quadro 5.6 nota-se a simplicidade em relação ao da etapa anterior, uma vez que somente é utilizada a metaclasse "Modelo". Isso se deve à existência do campo idModeloOrig, que relaciona o modelo tratado ao seu equivalente na etapa CIM. Com isso, não são necessárias informações dos requisitos que estão sendo tratados nesse momento. O Quadro 5.7 apresenta o modelo resultante com os elementos do modelo de rastreabilidade gerado no PIM.

Quadro 5.7: Modelo de Rastreabilidade PIM – Elementos.

Elemento						
Atributo do Metaelemento do R2MDD	IdElem	IdMod	nomeElemento	tipoElemento	idElemOrig	idPai
Elemento do Modelo de Rastreabilidade gerado no PIM	14	2	RecemNascido	Classe	2	*Null*
	15	2	CartaoRN	Classe	1	*null*
	16	2	Nome	Atributo	2	14
	17	2	nomeMae	Atributo	2	14
	18	2	dataNasc	Atributo	2	14
	19	2	Sexo	Atributo	2	14

	20	2	Gemelar	Atributo	2	14
	21	2	Endereço	Atributo	2	14
	22	2	Telefone	Atributo	2	14
	23	2	manterRecemNascido()	Funcao	2	14
	24	2	regSTRN	Atributo	3	15
	25	2	tipoCartao	Atributo	4	15
	26	2	postoColeta	Atributo	5	15
	27	2	dataColeta	Atributo	6	15
	28	2	Lote	Atributo	7	15
	29	2	Antibiótico	Atributo	8	15
	30	2	Prematuro	Atributo	9	15
	31	2	Transfusão	Atributo	10	15
	32	2	responsavelColeta	Atributo	11	15
	33	2	manterCartaoRN()	Funcao	12 e 13	15
	34	2	relRNCartaoRN	Relacionamento	1	*Null*

O Quadro 5.8 lista os rastros gerados com a criação do modelo de rastreabilidade do PIM.

Quadro 5.8: Modelo de Rastreabilidade PIM – Rastros.

Rastros							
Atributo do Metaelemento do R2MDD	**idRastro**	**Motivo**	**tipoRastro**	**idInfo**	**tipoInfo**	**data**	**responsavel**
Elemento do Modelo de Rastreabilidade gerado no PIM	17	Gerar modelo proveniente da transformação da etapa CIM para PIM com a inclusão de novo modelo de rastreabilidade na etapa PIM que atende ao requisito RF001	Inclusão	2	Modelo	17/04/16	izabella
	18	Criar a classe responsável pelas informações do RN	Inclusão	14	Elemento	17/04/16	izabella
	19	Criar classe responsável por manter informações do Cartão do RN	Inclusão	15	Elemento	17/04/16	izabella
	19	Armazenar o nome do RN	Inclusão	16	Elemento	17/04/16	izabella
	20	Armazenar nome da mãe do RN	Inclusão	17	Elemento	17/04/16	izabella
	21	Armazenar a data de Nascimento do RN	Inclusão	18	Elemento	17/04/16	izabella
	22	Indicar o Sexo do RN	Inclusão	19	Elemento	17/04/16	izabella
	23	Indicar se RN é irmão gêmeo	Inclusão	20	Elemento	17/04/16	izabella
	24	Armazenar o endereço do RN	Inclusão	21	Elemento	17/04/16	izabella
	25	Armazenar o telefone do RN	Inclusão	22	Elemento	17/04/16	izabella
	26	Manter as informações do RN	Inclusão	23	Elemento	17/04/16	izabella
	27	Armazenar valor do registro do STRN	Inclusão	24	Elemento	17/04/16	izabella
	28	Informar tipo de cartão	Inclusão	25	Elemento	17/04/16	izabella
	29	Armazenar o posto da coleta	Inclusão	26	Elemento	17/04/16	izabella

	30	Armazenar a data da coleta	Inclusão	27	Elemento	17/04/16	izabella
	31	Armazenar informação de lote	Inclusão	28	Elemento	17/04/16	izabella
	32	Indicar antibiótico utilizado	Inclusão	29	Elemento	17/04/16	izabella
	33	Indicar se paciente é prematuro	Inclusão	30	Elemento	17/04/16	izabella
	34	Indicar se é necessária transfusão	Inclusão	31	Elemento	17/04/16	izabella
	35	Armazenar o responsável pela coleta	Inclusão	32	Elemento	17/04/16	izabella
	36	Manter cartão do RN	Inclusão	33	Elemento	17/04/16	izabella
	37	Relacionar classes RescemNascido e CartaoRN	Inclusão	34	Elemento	17/04/16	izabella

O modelo gerado na etapa PIM permite a identificação do modelo que o originou, ligado ao CIM e as respostas das seguintes perguntas:

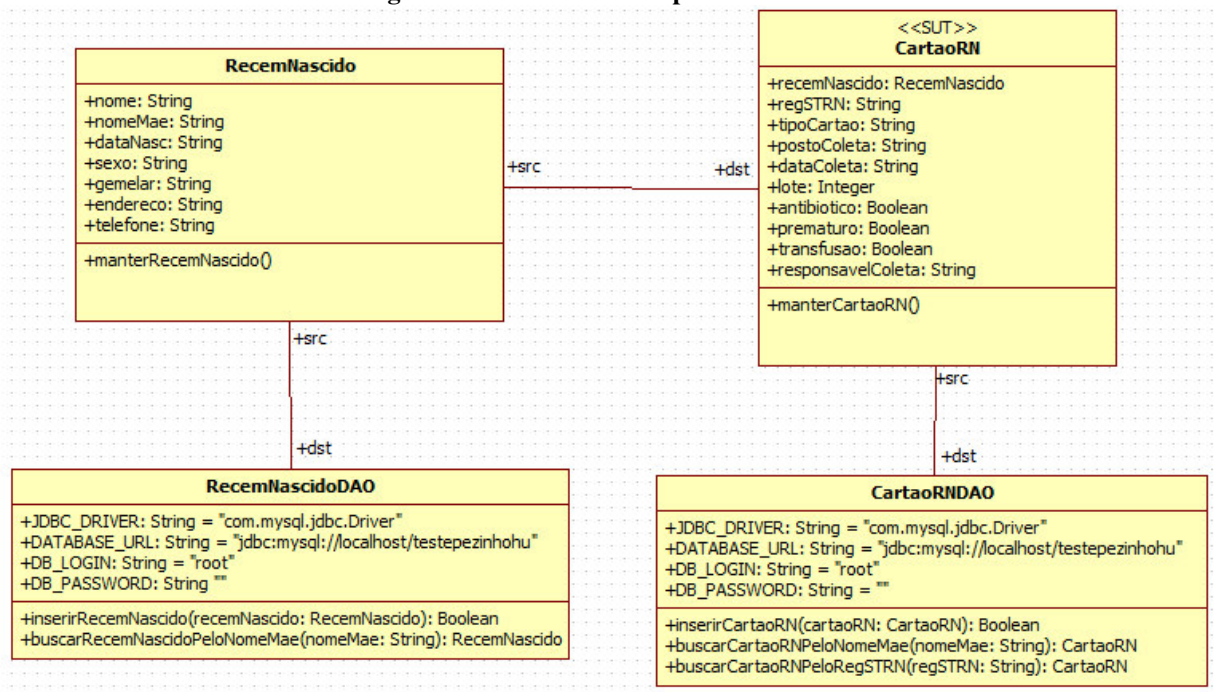
- O Que? Um rastro foi gerado após a transformação de modelos da etapa CIM para PIM;
- Quem? O "responsável" foi "izabella";
- Como? Por meio da "Inclusão" de um novo "Modelo" ("M002") na etapa PIM, que está diretamente ligado ao "Modelo" gerado na etapa CIM. Essas informações podem ser identificadas através dos campos: idInfo (preenchido com 2), e tipoInfo (preenchido com "Modelo"). Ao identificar qual modelo está ligado ao rastro, é possível saber qual foi o modelo que originou por meio do atributo idModeloOrig ("M001") e, nele, é possível identificar qual o requisito que está sendo tratado, por meio da classe associativa "RequisitoModelo";
- Onde? Na etapa PIM, onde foram incluídas informações dos modelos gerados e elementos associados;
- Porque? Para gerar modelo proveniente da transformação da etapa CIM para PIM com a inclusão de novo modelo de rastreabilidade na etapa PIM que atende ao requisito RF001; e,
- Quando? Em 17/04/16.

Com isso, o modelo gerado na etapa PIM atende aos requisitos da rastreabilidade, por responder a todas as perguntas relacionadas aos rastros.

5.3.3 Etapa PSM

Nessa etapa, foi considerada que, para a aplicação, será utilizada a linguagem de programação Java. Nesse sentido, os elementos de origem serão mapeados para os de destino gerando o diagrama ilustrados pela Figura 5.6.

Figura 5.5: Modelo PSM a partir do PIM.



Fonte: (ALMEIDA, 2014).

Nos Quadros 5.9, 5.10 e 5.11, o resultado do modelo de rastreabilidade gerado para essa etapa é apresentado. Como, nesse projeto, foi considerado somente a linguagem de programação Java, foi gerado apenas um modelo para o PSM. Entretanto, caso sejam necessárias outras tecnologias, outros modelos devem ser gerados.

Quadro 5.9: Modelo de Rastreabilidade PSM.

Metaelemento do R2MDD	Atributo do Metaelemento do R2MDD	Elemento do Modelo de Rastreabilidade gerado no PSM
Modelo	idMod	3
	codModelo	M003
	descricaoModelo	Primeiro modelo de rastreabilidade gerado no PSM.
	situacaoModelo	Ativo
	Etapa	PSM
	idModOrig	2

Quadro 5.10: Modelo de Rastreabilidade PSM – Elementos.

Elemento						
Atributo do Metaelemento do R2MDD	**idElem**	**IdMod**	**nomeElem**	**tpElem**	**idElemOrig**	**idPai**
Elemento do Modelo de Rastreabilidade gerado no PSM	36	3	RecemNascido	Classe	14	*Null*
	37	3	CartaoRN	Classe	15	*Null*
	38	3	nome	Atributo	16	36
	39	3	nomeMae	Atributo	17	36
	40	3	dataNasc	Atributo	18	36
	41	3	sexo	Atributo	19	36
	42	3	gemelar	Atributo	20	36
	43	3	endereco	Atributo	21	36
	44	3	telefone	Atributo	22	36
	45	3	manterRecemNascido()	Função	23	36
	46	3	regSTRN	Atributo	24	37
	47	3	tipoCartao	Atributo	25	37
	48	3	postoColeta	Atributo	26	37
	49	3	dataColeta	Atributo	27	37
	50	3	Lote	Atributo	28	37
	51	3	antibiotico	Atributo	29	37
	52	3	prematuro	Atributo	30	37
	53	3	transfusao	Atributo	31	37
	54	3	responsavel Coleta	Atributo	32	37
	55	3	manter CartaoRN()	Função	33	37
	56	3	relRN CartaoRN	Relacionamento	34	*Null*
	57	3	RecemNascidoDAO	Classe	14	*Null*
Elemento do Modelo de Rastreabilidade gerado no PSM	58	3	JDBC_Driver	Atributo	*Null*	60
	59	3	DATABASE_URL	Atributo	*Null*	60
	60	3	DB_LOGIN	Atributo	*Null*	60
	61	3	DB_PASSWORD	Atributo	*Null*	60
	62	3	inserirRecemNascido (recemNascido: RecemNascido)	Atributo	23	60
	63	3	buscarRecem NascidoPelo NomeMae (nomeMae:String)	Atributo	23	60
	64	3	relRecemNascidoDAO	Relacionamento	34	*Null*
	65	3	CartaoRNDAO	Classe	15	*Null*
	66	3	JDBC_Driver	Atributo	*Null*	67
	67	3	DATABASE_URL	Atributo	*Null*	67
	68	3	DB_LOGIN	Atributo	*Null*	67
	69	3	DB_PASSWORD	Atributo	*Null*	67
	70	3	inserirCartaoRN(cartaoR N CartaoRN)	Função	33	67
	71	3	buscarCartaoRNpeloNo meMae(nomeMae: String)	Função	33	67
	72	3	buscarCartaoRNpeloReg STRN (regSTRN: String)	Função	33	67
	73	3	relCartaoRN_DAO	Relacionamento	34	*Null*

Quadro 5.11: Modelo de Rastreabilidade PSM – Rastros.

Rastros							
Atributo do Metaelemento do R2MDD	**idRastro**	**Motivo**	**tipoRastro**	**idInfo**	**tipoInfo**	**data**	**responsavel**
Elemento do Modelo de Rastreabilidade gerado no PIM	40	Indicar transformação de modelos da etapa PIM para PSM com a inclusão de novo modelo na etapa PSM que atende ao requisito RF001	Inclusão	3	Modelo	17/04/16	izabella
	41	Criar a classe responsável pelas informações do RN	Inclusão	36	Elemento	17/04/16	izabella
	42	Criar classe responsável por manter informações do Cartão do RN	Inclusão	37	Elemento	17/04/16	izabella
	44	Armazenar o nome do RN	Inclusão	38	Elemento	17/04/16	izabella
	45	Armazenar nome da mãe do RN	Inclusão	39	Elemento	17/04/16	izabella
	46	Armazenar a data de Nascimento do RN	Inclusão	40	Elemento	17/04/16	izabella
	47	Indicar o Sexo do RN	Inclusão	41	Elemento	17/04/16	izabella
	48	Indicar se RN é irmão gêmeo	Inclusão	42	Elemento	17/04/16	izabella
	49	Armazenar o endereço do RN	Inclusão	43	Elemento	17/04/16	izabella
	50	Armazenar o telefone do RN	Inclusão	44	Elemento	17/04/16	izabella
	51	Manter as informações do RN	Inclusão	45	Elemento	17/04/16	izabella
	52	Armazenar valor do registro do STRN	Inclusão	46	Elemento	17/04/16	izabella
	53	Informar tipo de cartão	Inclusão	47	Elemento	17/04/16	izabella
	54	Armazenar o posto da coleta	Inclusão	48	Elemento	17/04/16	izabella
	55	Armazenar a data da coleta	Inclusão	49	Elemento	17/04/16	izabella
	56	Armazenar informação de lote	Inclusão	50	Elemento	17/04/16	izabella
	57	Indicar antibiótico utilizado	Inclusão	51	Elemento	17/04/16	izabella
	58	Indicar se paciente é prematuro	Inclusão	52	Elemento	17/04/16	izabella
	59	Indicar se é necessária transfusão	Inclusão	53	Elemento	17/04/16	izabella
	60	Armazenar o responsável pela coleta	Inclusão	54	Elemento	17/04/16	izabella
	61	Manter cartão do RN	Inclusão	55	Elemento	17/04/16	izabella
	62	Relacionar classes RescemNascido e	Inclusão	56	Elemento	17/04/16	izabella

		CartaoRN					
	63	Inclusão da classe RecemNascido DAO	Inclusão	57	Elemento	17/04/16	izabella
	64	Inclusão do atributo JDBC_Driver	Inclusão	58	Elemento	17/04/16	izabella
	65	Inclusão do atributo DATABASE_ URL	Inclusão	59	Elemento	17/04/16	izabella
	66	Inclusão do atributo DB_LOGIN	Inclusão	60	Elemento	17/04/16	izabella
	67	Inclusão do atributo DB_ PASSWORD	Inclusão	61	Elemento	17/04/16	izabella
	68	Inclusão da Função inserirRecem Nascido (recemNascido: RecemNascido)	Inclusão	62	Elemento	17/04/16	izabella
	69	Inclusão da Função buscarRecem NascidoPelo NomeMae (nomeMae:String)	Inclusão	63	Elemento	17/04/16	izabella
Elemento do Modelo de Rastreabilidade gerado no PIM	70	Inclusão do relacionamento relRecem NascidoDAO	Inclusão	64	Elemento	17/04/16	izabella
	71	Inclusão da Classe CartaoRNDAO	Inclusão	65	Elemento	17/04/16	izabella
	72	Inclusão do atributo JDBC_Driver	Inclusão	66	Elemento	17/04/16	izabella
	73	Inclusão do atributo DATABASE_URL	Inclusão	67	Elemento	17/04/16	izabella
	74	Inclusão do atributo DB_LOGIN	Inclusão	68	Elemento	17/04/16	izabella
	75	Inclusão do atributo DB_ PASSWORD	Inclusão	69	Elemento	17/04/16	izabella
	76	Inclusão da função inserirCartaoRN (cartaoRN CartaoRN)	Inclusão	70	Elemento	17/04/16	izabella
	77	Inclusão da função buscarCartao RNpeloNomeMae (nomeMae: String)	Inclusão	71	Elemento	17/04/16	izabella
	78	Inclusão da função buscarCartao RNpeloReg STRN(regSTRN: String)	Inclusão	72	Elemento	17/04/16	izabella
	79	Inclusão do relacionamento relCartaoRN_DAO	Inclusão	73	Elemento	17/04/16	izabella

O modelo gerado na etapa PSM responde às seguintes perguntas:

- O Que? Um rastro foi gerado após a transformação de modelos da etapa PIM para PSM;
- Quem? O "responsável" foi "izabella";
- Como? Por meio da "Inclusão" de um novo "Modelo" ("M003") na etapa PSM, que está diretamente ligado ao "Modelo" gerado na etapa PIM. Essas informações podem ser identificadas através dos campos: idInfo (preenchido com 3), e tipoInfo (preenchido com "Modelo"). Ao identificar qual modelo está ligado ao rastro, é possível saber qual foi o modelo que o originou na etapa anterior por meio do atributo idModeloOrig ("M002"), e, com isso também identificar o modelo do CIM ("M001");
- Onde? Na etapa PSM, onde foram incluídas informações dos modelos gerados e elementos associados;
- Porque? Para indicar transformação de modelos da etapa PIM para PSM com a inclusão de novo modelo na etapa PSM que atende ao requisito RF001; e,
- Quando? Em 17/04/16.

Com isso, o modelo gerado na etapa PSM continua a atender aos requisitos da rastreabilidade, por responder a todas as perguntas relacionadas aos rastros.

5.3.4 Etapa Código

Nessa última etapa, ocorreu a transformação do modelo PSM para o código fonte do projeto. Uma parte do código gerado pode ser vista na Figura 5.7.

Figura 5.6: Código a partir do PSM.

```
18
19
20
21
22  public Boolean inserirCartaoRN(CartaoRN cartaoRN) {
23      Connection conn;
24      try {
25          Class.forName(JDBC_DRIVER);
26          conn = DriverManager.getConnection(DATABASE_URL, DB_LOGIN, DB_PASSWORD);
27          Statement st = conn.createStatement();
28          int prematuro = (cartaoRN.isPrematuro()) ? 1 : 0;
29          int transfusao = (cartaoRN.isTransfusao()) ? 1 : 0;
30          int antibiotico = (cartaoRN.isAntibiotico()) ? 1 : 0;
31          String query = "INSERT INTO cartao_recem_nascido (recem_nascido_fk,reg_strn,tipo_cartao,posto_coleta,"
32                  +"data_coleta,lote,antibiotico,prematuro,transfusao,responsavel_coleta) VALUES "
33                  +"('"+cartaoRN.getRecemNascido().getId()+"','"+cartaoRN.getRegSTRN()+"','"+cartaoRN.getTipoCartao()+"',"
34                  +"'"+cartaoRN.getPostoColeta()+"','"+cartaoRN.getDataColeta()+"','"+cartaoRN.getLote()+"','"+antibiotico+"',"
35                  +"'"+prematuro+"','"+transfusao+"','"+cartaoRN.getResponsavelColeta()+"')";
36          st.executeUpdate(query);
37          conn.close();
38          st.close();
39          return true;
40      } catch (SQLException s) {
41          s.printStackTrace();
42      } catch (Exception e) {
43          e.printStackTrace();
44      }
45      return false;
46  }
```

Fonte: (ALMEIDA, 2014).

Para representar as informações da Figura 5.6, o modelo de rastreabilidade gerado é representado nos Quadros 5.12, 5.13 e 5.14.

Quadro 5.12: Modelo de Rastreabilidade – Código.

Metaelemento do R2MDD	Atributo do Metaelemento do R2MDD	Elemento do Modelo de Rastreabilidade gerado no Código
Modelo	idMod	4
	codModelo	M004
	descricaoModelo	Primeiro modelo de rastreabilidade gerado no Código.
	situacaoModelo	Ativo
	Etapa	Código
	idModOrig	3

Quadro 5.13: Modelo de Rastreabilidade Código – Elementos.

	Elemento					
Atributo do Metaelemento do R2MDD	idElem	IdMod	nomeElem	tpElem	idElemOrig	idPai
Elemento do Modelo de Rastreabilidade gerado no CIM	74	3	inserirCartaoRN(CartaoRN cartaoRN)	Método	62	
	75	3	CartaoRN	Classe	41	
	76	3	cartaoRN	Campo	58	75
	77	3	Prematuro	Campo	58	75
	78	3	cartaoRN.isPrematuro()	Comando	59	75
	79	3	Transfusão	Campo	59	75
	80	3	cartaoRN.isTransfusao()	Comando	57	75

	81	3	Antibiótico	Campo	61	75
	82	3	cartaoRN.isAntibiotico()	Comando	61	75
	83	3	cartaoRN.getRecemNascido().getID()	Comando	61	75
	84	3	cartaoRN.getRegSTRN()	Comando	61	75
	85	3	cartaoRN.getTipoCartao()	Comando	61	75
	86	3	cartaoRN.getPostoColeta()	Comando	61	75
	87	3	cartaoRN.getDataColeta()	Comando	61	75
	88	3	cartaoRN.getLote()	Comando	61	75
	89	3	cartaoRN.getResponsavelColeta()	Comando	61	75

Quadro 5.14: Modelo de Rastreabilidade Código – Rastros.

Rastros							
Atributo do Metaelemento do R2MDD	**idRastro**	**Motivo**	**tipoRastro**	**idInfo**	**tipoInfo**	**data**	**responsavel**
Elemento do Modelo de Rastreabilidade gerado no PIM	80	Indicar transformação de modelos da etapa PSM para Código com a inclusão de novo modelo na etapa Código que atende ao requisito RF001	Inclusão	3	Modelo	17/04/16	izabella
	81	Criar classe responsável por manter informações do cartão do RN	Inclusão	75	Elemento	17/04/16	izabella
	82	Criar método inserirCartaoRN que faz a inclusão de registros do cartão do RN	Inclusão	74	Elemento	17/04/16	izabella
	83	Criar instância da Classe CartaoRN para manipular valores	Inclusão	76	Elemento	17/04/16	izabella
	84	Criar campo prematuro que obterá informações da chamada do comando cartaoRN.isPrematuro()	Inclusão	77	Elemento	17/04/16	izabella

O modelo gerado na etapa Código responde às seguintes perguntas:

- O Que? Um rastro foi gerado após a transformação de modelos da etapa PSM para Código;
- Quem? O "responsável" foi "izabella";
- Como? Através da inclusão do novo modelo M004 resultante da transformação do PSM para Código.
- Onde? Na etapa Código.

- Porque? Para indicar transformação de modelos da etapa PSM para Código com a inclusão de novo modelo na etapa Código que atende ao requisito RF001; e,
- Quando? Em 17/04/16.

Com isso, o modelo gerado na etapa Código continua a atender aos requisitos da rastreabilidade, por responder a todas as perguntas relacionadas aos rastros.

5.3.5 Monitoramento de Requisitos

O nível 04 do R2MDD, referente ao monitoramento de requisitos, trata do acompanhamento dos requisitos, com a possível geração de diversos tipos de relatórios ou até mesmo modelos, com informações dos modelos de rastreabilidade. O relatório de impacto de alteração de elementos com as informações dos modelos gerados nas etapas CIM, PIM, PSM e Código é ilustrado na Figura 5.5.

Figura 5.5: Relatório de Impactos de Alteração de Elementos.

Relatório de Impacto de Alteração de Elementos		
Requisito:	RF004	
Elemento:	transfusão	
Tipo Alteraç	Exclusão	
Motivo:	Não é mais necessário o atributo transfusão	
Data:	01/05/2016	
Elementos que serão afetados		
Etapa	**Local Afetado**	**Nome**
CIM	Requisito	RF004
	Modelo	M001
	Elemento - Atributo	transfusão
	Elemento - Classe	CartaoRN()
PIM	Requisito	RF004
	Modelo	M002
	Elemento - Atributo	transfusão
	Elemento - Classe	CartaoRN()
PSM	Requisito	RF004
	Modelo	M003
	Elemento - Atributo	transfusão
	Elemento - Classe	CartaoRN()
Código	Requisito	RF004
	Modelo	M004
	Elemento - Classe	CartaoRN()
	Elemento - Campo	transfusao
	Elemento - Comando	CartaoRN.isTransfusao()
	Elemento - Campo	query

É possível observar todos os elementos, modelos e requisitos que serão afetados caso exista a necessidade de se excluir o elemento “transfusão” do modelo em cada fase da MDA.

Outra forma de identificar informações dos requisitos é por meio da matriz de rastreabilidade, um exemplo é apresentado na Figura 5.6.

Figura 5.6: Matriz de Rastreabilidade STRN.

Na Figura 5.6, é possível visualizar que existe uma relação de satisfação de RF002 com RF004, identificada na Etapa CIM.

5.4 Avaliação do R2MDD

Durante a execução do experimento, algumas considerações gerais foram coletadas com o propósito de refinamento do *framework*. Os pontos destacados foram:

- O estado dos requisitos e seus artefatos durante a execução das transformações da MDA pode ser identificado de forma fácil através dos relatórios gerados no Nível 04;
- É possível identificar durante o Nível 03, quais artefatos estão alocados a requisitos e modelos e como eles se relacionam;
- O metamodelo de apoio ao R2MDD responde a todas as dimensões da rastreabilidade em todas as etapas;

5.5 Considerações Finais do Capítulo

Esse capítulo apresentou o experimento utilizado para validar o R2MDD , descrevendo suas etapas e, ao final, discutindo os resultados obtidos. Para realizar o

experimento desta pesquisa foi utilizado o modelo *Qualitas* proposto por Almeida (2014), o qual busca orientar projetos de *software* orientado a modelos.

O objeto de estudo desta pesquisa foi o o PNTN do HU-UFS, com foco no requisito funcional RF004, levantado durante a validação das etapas de planejamento do projeto, planejamento da execução e engenharia de requisitos do *Qualitas*, por Almeida (2014).

Durante as transformações entre modelos, realizadas nas etapas CIM, PIM, PSM e Código, foram gerados, de forma manual, modelos de rastreabilidade com informações referentes aos modelos da MDA. Esses modelos puderam ser utilizados como base para o monitoramento dos requisitos em todo o processo.

Através da análise do experimento, foi possível observar que o *framework* R2MDD proporciona o acompanhamento dos requisitos e artefatos durante os processos do MDD, promovendo maior eficácia na ER. Ele também proporciona aos gerentes e *Stakeholders* os benefícios obtidos com a rastreabilidade de requisitos, como análises de impactos, validação de requisitos, possibilidade de previsão de custos e tempo, entre outros.

Algumas limitações foram identificadas, como a inexistência de uma ferramenta de apoio para registrar os modelos e fazer transformações de modelos gerados na MDA para os modelos de rastreabilidade, fazendo com que todo o processo de validação tenha ocorrido de maneira manual.

Desta forma, propõe-se que sejam realizados mais estudos a fim de promover o aprimoramento do R2MDD e suas heurísticas, visando a melhoria do *framework* e facilidade de execução em um ambiente real.

No capítulo 6 serão feitas as considerações finais do trabalho, apresentando os principais resultados, limitações da pesquisa e trabalhos futuros.

CAPÍTULO 6

CONSIDERAÇÕES FINAIS

Alguns dos grandes desafios nos projetos de *software* estão relacionados a redução de custos e ao cumprimento dos prazos, bem como a fazer com que o produto a ser desenvolvido atenda a todas as necessidades dos clientes. As abordagens dirigidas a modelos fazem parte desse cenário, visando buscar soluções para esses desafios. Elas sugerem uma importante mudança de paradigma no qual o artefato principal do desenvolvimento passa a ser o modelo.

O MDD é uma abordagem que ganhou destaque por possuir uma perspectiva voltada para modelos no processo de desenvolvimento. Uma série de benefícios podem ser gerados com a utilização do MDD, como a melhoria e cumprimento dos prazos, facilidade de manutenção, a redução dos custos de desenvolvimento, melhoria da consistência de *software*, qualidade e geração de código-fonte através de modelos. Uma de suas principais ideias é a transformação de modelos de níveis mais altos de abstração para os níveis mais baixos com o objetivo de geração automática de código fonte.

A utilização do MDD, entretanto, possui algumas limitações. Uma delas está relacionada ao rastreamento e monitoramento dos requisitos durante todo o processo de transformação de modelos, uma vez que ainda não é dado um *feedback* de maneira clara sobre como os requisitos se comportam ao longo das transformações.

Para auxiliar na resolução do problema descrito acima, torna-se importante alinhar as práticas da ER ao MDD. A ER é a área da Engenharia de *Software* que trata dos requisitos. É por meio dela que é possível levantar, analisar, negociar, documentar, validar e gerir os requisitos. O controle e o monitoramento dos requisitos podem ser realizados por meio da rastreabilidade de requisitos, etapa do processo da ER referente ao gerenciamento de requisitos.

A rastreabilidade de requisitos, então, pode ser um meio eficaz para resolver uma série de problemas que ocorrem devido à falta de acompanhamento dos requisitos durante os processos do MDD.

Neste trabalho apresentou-se o R2MDD, que visa integrar a rastreabilidade e o monitoramento de requisitos ao processo do MDD. O *framework* tem o objetivo de facilitar o acompanhamento dos requisitos gerando diversos benefícios aos envolvidos no processo de desenvolvimento e demais *stakeholders*.

O R2MDD está organizado em quatro níveis: metamodelo de rastreabilidade de requisitos, modelo de rastreabilidade, MDA e monitoramento de requisitos. Ele consiste na geração de modelos de rastreabilidade que devem estar em conformidade com um metamodelo durante as etapas da MDA. Durante o processo, podem ser gerados diversos relatórios de monitoramento de requisitos com o intuito de identificar o estado de um requisito ou elemento em determinado momento.

Para validar o R2MDD, foi realizado um caso exemplo com o auxílio do modelo de processo *Qualitas*, proposto por Almeida (2013), e aplicado no Sistema de Triagem NeoNatal do Hospital Universitário de Sergipe.

O caso exemplo foi muito significativo para determinar alguns pontos fortes e chegar a algumas conclusões. Entre os resultados observados estão:

- É um *framework* de fácil entendimento, enxuto e com poucos níveis;
- Os modelos são o centro do processo;
- A rastreabilidade também pode ser definida a partir da geração de modelos próprios;
- Oferece a possibilidade de se associar a outros *frameworks*, modelos e metodologias, como, por exemplo, o próprio modelo *Qualitas*, o *Scrum*, entre outros; e,
- Durante todo o processo da MDA podem ser acompanhados os estados de um determinado requisito ou elemento.

6.1 Principais Contribuições

As principais contribuições deste trabalho são:

- Identificação do estado da arte das Abordagens Dirigidas a Modelos, com foco em MDD no Capítulo 2;
- Análise da importância dos requisitos em MDD na seção 2.5;
- Identificação do estado da arte da ER, rastreabilidade e monitoramento de requisitos no Capítulo 3;

- Análise de metamodelos de rastreabilidade de requisitos na seção 3.4;
- Análise do estado da arte da rastreabilidade de requisitos em MDD por meio de uma revisão sistemática, na seção 3.5;
- Definição do *framework* R2MDD;
- Validação do R2MDD por meio de um experimento com auxílio do Modelo *Qualitas*; e,
- Publicação de uma revisão sistemática no Congresso Internacional de Gestão de Tecnologia e Sistemas de Informação (2015).

6.2 Limitações da Pesquisa

Ao desenvolver esta pesquisa, foram encontradas algumas limitações, sendo elas:

- Ausência de ferramentas que suportassem a aplicação do R2MDD, e, por isso, a necessidade de construção dos modelos de rastreabilidade durante o caso exemplo de forma totalmente manual;
- Foi utilizado somente um caso exemplo, entretanto, para a validação efetiva são necessárias aplicações em diversos casos.

6.3 Trabalhos Futuros

Para resolver algumas dessas limitações, foram identificados possíveis trabalhos futuros:

- Realização de novos casos exemplos e estudos experimentais para validação do R2MDD;
- Aperfeiçoamento do R2MDD, após análise e realização de novos estudos experimentais;
- Desenvolvimento de um *plugin* que gere os modelos de forma automática e também que auxilie no monitoramento dos requisitos;
- Disponibilização desse *plugin* para a comunidade acadêmica através de ambientes amplamente conhecidos, como o Portal do *Software* Público, de forma a favorecer a disseminação das técnicas utilizadas, bem como discussões e melhorias do R2MDD por outros pesquisadores; e,

- Publicações dos resultados dos estudos em conferências e periódicos a fim de compartilhar conhecimento, bem como buscar discussões e sugestões de melhorias na comunidade acadêmica.

REFERÊNCIAS

ABNT, Associação Brasileira de Normas Técnicas. Engenharia de Software – Qualidade do Produto. Parte 1: Modelo de Qualidade. NBR ISO/IEC 9126-1. 2003. Disponível em: < http://goo.gl/ZOcGHo>. Acesso em 15/01/2016.

ALMEIDA, C. C. J. *Qualitas:* Um Modelo de Processo de Desenvolvimento Orientado a Modelos. 2014.

ALMEIDA, J. P. A; IACOB, M. E.; VAN ECK, P. Requirements traceability in model-driven development: Applying model and transformation conformance. Information Systems Frontiers, v. 9, n. 4, p. 327–342, 2007.

AMELLER, D. Non-Functional Requirements as drivers of Software Architecture Design. Tese de Doutorado. Universitat Politècnica de Catalunya. 2009.

_____. Dealing with Non-Functional Requirements in Model-Driven Development. Publicado em Requirements Engineering Conference (RE), 2010 18th IEEE International. 2010.

ASADI, M.; ESFAHANI, N.; RAMSIN, R. Process patterns for MDA-based software development. 8th ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2010, p. 190–197, 2010.

ATKINSON, C.; KÜHNE, T. Model-driven development: A metamodeling foundation. IEEE Software, v. 20, n. 5, p. 36–41, 2003.

AUTUMN. Software Requirements. CS2 Software Engineering note 2. CS2Ah, 2004. Disponível em: < http://goo.gl/DlMBtH>. Acesso em 15/01/2016.

AZOFF, M. The Benefits of Model Driven Development. Butter Group´s White Paper, 2008.

BEREZIN, T. Writing a Requirements Document. 1999. Disponível em: < http://goo.gl/ydbXa4>. Acesso em 15/01/2016.

BATISTA, S.Uma Ferramenta de Apoio a Definição de Requisitosda MDSODI no Contexto do Ambiente DiSEN. Dissertação de Mestrado. Universidade Federal do Paraná. Curitiba, 2003. Disponível em: < http://goo.gl/uUu1Og>. Acesso em 15/01/2016.

BÉZIVIN, J. History and Context of MDE. Principles and Applications of Model Driven Engineering. Disponível em: <http://www.nii.ac.jp/userimg/lectures/20120117/Lecture1.pdf>. Acesso em 15/04/2015. 2014.

BOEHM, B. A view of 20th and 21st century software engineering. Proceeding of the 28th international conference on Software engineering - ICSE '06, p. 12, 2006.

CALINESCU, R. C. Methodology for the model-driven development of self-managing systems. p. 60558, 2008.

CAMPOS, P. Engenharia de Requisitos. Cee.Uma.Pt, 2013.

CASTRO, J. O processo da Engenharia de Requisitos. 2005. Disponível em: <http://goo.gl/cwXjNY>. Acesso em 15/12/2015.

CERNOSEK, G.; XDE, R. R. development. n. December, 2004.

CHITFOROUSH, F.; YAZDANDOOST, M.; RAMSIN, R. Methodology Support for the Model Driven Architecture. 14th Asia-Pacific Software Engineering Conference (APSEC'07), p. 454–461, 2007.

COMMITTEE, S. E. S. IEEE recommended practice for software requirements specifications. [s.l: s.n.]. v. 1998.

DRAFFIN, A., Methodology vs framework – why waterfall and agile are not methodologies. 2007. Disponível em <https://goo.gl/gHNw8W>.

DEMIR, A. Comparison of model-driven architecture and software factories in the context of model-driven development. Proc. - Joint Meeting of the 4th Workshop on Model-Based Dev. of Computer-Based Systems and the 3rd Int. Workshop on Model-Based Methodologies for Pervasive and Embedded Software, MBD/MOMPES 2006, p. 75–83, 2006.

FALBO, R. A. Engenharia de Requisitos. Universidade Federal do Espírito Santo, Vitória, 2012. Disponível em: <http://goo.gl/yBZY3v>.

FERNANDES, L.; FERNANDES, L. C&L: Uma Ferramenta de Apoio à Engenharia de Requisitos. Requirements Engineering, 2004.

FRANCE, R.; RUMPE, B. Model-driven Development of Complex Software: A Research Roadmap. Future of Software Engineering (FOSE '07), n. 2, 2007.

FRANCETO, S. Especificação e Implementação de uma Ferramenta para Elicitação de Requisitos de Software baseada na teoria da Atividade. Programa de Pós-Graduação em Ciência da Computação. Piracicaba, SP. 2005.

FRANCH, X.; AMELLER, D.; CABOT, J. Dealing with non-functional requirements in model-driven development. Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, RE2010, p. 189–198, 2010.

GENVIGIR, E.; VIJAYKUMAR, N. Uma Proposta de Modelagem para Generalização de Elos de rastreabilidade. Revista de Informática Teórica e Aplicada, Volume XV, n. 2. 2008.

GHARAAT, A.; TSUI, F.; DUGGINS, S.; JUNG, E. Measuring Levels of Abstraction in Software Development. 23rd International Conference on Software Engineering & Knowledge Engineering, p. 466–469, 2011.

GONÇALVES, A. et al. IEEE Std 830 Prática Recomendada Para Especificações de Exigências de Software: Standard Internacional. p. 38, 2004.

GOTEL, O. C.; FINKELSTEIN, A. C. W.; SW, L. An Analysis of the Requirements Traceability Problem Imperial College of Science, Technology & Medicine Department of Computing, 180 Queen' s Gate. p. 94–101, 1994.

GRILLO, F. Uma Ferramenta acessível de apoio à modelagem de software na Web. Exame de qualificação de mestrado. Disponível em < http://goo.gl/uWtQMt>. Acesso em 15/04/2015. 2012.

GUSTAVO, A. Uma Abordagem Ágil e Dirigida por Modelos para Desenvolvimento de Software Embarcados e de Tempo-Real.pdf. 2010.

HERRERA, F. et al. A model-driven methodology for the development of SystemC executable environments. Specification and Design Languages (FDL), 2012 Forum on, p. 177–184, [s.d.].

HOYOS, H. et al. HiLeS2: model driven embedded system virtual prototype generation. 2011 Theory of Modeling & Simulation Symposium: DEVS Integrative M&S Symposium, p. 75–82, 2011.

HULL, E.; JACKSON, K.; DICK, J. Requirements Engineering. 2 Springer London Berlin Heidelberg. v. 13. 2004. Disponível em <http://goo.gl/Lvo02c>. Acesso em 14/04/2015.

HU-UFS. Hospital Universitário da Universidade Federal de Sergipe. Disponível em: <http://www.ebserh.gov.br/web/hu-ufs> . Acesso em 16/03/2016.

KILICAY-ERGIN, N.; LAPLANTE, P. A. An online graduate requirements engineering course. IEEE Transactions on Education, v. 56, n. 2, p. 208–216, 2013.

KOTONYA, G.; SOMERVILLE, I. Requirements engineering: processes and techniques. p. 25–52, 1998.

_____.KOTONYA, G.; SOMMERVILLE, I. Requirements Engineering – Processes and Techniques. John Willy & Sons, 1997.

KOVAČEVIĆ, J., AFÉREZ, M., KULESZA, U., MOREIRA, A., ARAÚJO, J., AMARAL, V. (2007). Survey of the state-of-the-art in Requirements Engineering for Software Product Line and Model-Driven Requirements Engineering. AMPLE Deliverable D1.1. 2007. Disponível em: < http://ample.holos.pt/gest_cnt_upload/editor/File/public/Deliverable%20D1.1.pdf> Acesso em 15/01/2016.

KRAMER, J. Abstraction in Computer Science & Software Engineering: A Pedagogical Perspective. System Design Frontier Journal, v. 3, n. 12, p. 1–9, 2006.

KRAMER, J.; HAZZAN, O. The Role of Abstraction in Software Engineering. ACM SIGSOFT Software Engineering Notes, v. 31, n. 6, p. 38–39, 2006.

LAKATOS, E.; MARCONI, M. D. A. Metodologia científica. Metodologia Científica, p. 1–20, 1991.

LAZARTE, I. M. et al. Model-driven development methodology for B2B collaborations. Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC, p. 69–78, 2010.

LEAL, J. RAISE: Um metamodelo de informação de rastreabilidade. Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais. 2011.

LEITE, J. Análise e especificação de requisites. 1999. Disponível em: <https://goo.gl/kSn5Sh>. Acesso em: <15/04/2015>.

LONIEWSKI, G.; ARMESTO, A.; INSFRAN, E. An architecture-oriented model-driven requirements engineering approach. 2011 Model-Driven Requirements Engineering Workshop, MoDRE 2011, n. Cim, p. 31–38, 2011.

LUCREDIO, D. Uma Abordagem Orientada a Modelos para Reutilização de Software. PhD thesis, Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação. Disponível em: < http://goo.gl/sAQif2>. Acesso em 14/04/2015. 2009.

LUCRÉDIO, D.; GRILLO, F.; FORTES, F.; Towards collaboration between sighted and visually impaired developers in the context of Model-Driven-Engineering. Instituto de Matemática e Ciência da Computação da Universidade de São Paulo. Disponivel em < http://goo.gl/EqxCnR>. Acesso em 15/04/2015. 2012.

MACDONALD, A.; RUSSELL, D.; ATCHISON, B. Model-driven development within a legacy system: An industry experience report. Proceedings of the Australian Software Engineering Conference, ASWEC, v. 2005, n. Mdd, p. 14–22, 2005.

MACIEL, L. Uma abordagem dirigida por modelos para Geração Automática de Casos de teste de Integração Usando Padrões de Teste. Dissertação de Mestrado. Universidade Federal de Campina Grande. 2010.

MARTIN, S. et al. Requirements engineering process models in practice. … in Requirements Engineering (…, p. 141–155, 2002.

MELLO, L; MEYER, J. Levantamento de Requisitos. Disponível em: <http://goo.gl/pW5Iy1>. Acesso em 15/04/2015.

MELLOR, S. J. et al. Model-Driven Development. IEEE Software, v. 20, p. 14–18, 2003.

MERILINNA, J. YRJONEN, A**.** Tooling for the full traceability of non-functional requirements within model-driven development. Publicado em ECMFA-TW '10. Proceedings of the 6th ECMFA Traceability Workshop. 2010.

MINISTÉRIO DA SAÚDE. Manual de Normas Técnicas e Rotinas Operacionais do Programa Nacional de Triagem Neonatal. 2002. Brasília – DF. Disponível em <http://goo.gl/QE4ZnT>. Acesso em 15/01/2016.

NIKIFOROVA, O.; CERNICKINS, A.; PAVLOVA, N. Discussing the difference between model driven architecture and model driven development in the context of supporting tools the projection of two-hemisphere model into the component model of MDA/MDD. 4th International Conference on Software Engineering Advances, ICSEA 2009, Includes SEDES 2009: Simposio para Estudantes de Doutoramento em Engenharia de Software, p. 446–451, 2009.

OLIVEIRA, P.; PAIVA, T. Rastreamento de Requisitos. Dissertação de Mestrado, Universidade Federal de Pernambuco. 2009. Disponível em: < http://goo.gl/VfQgKQ> . Acesso em 15/01/2016.

OMG. Object Management Group, Model Driven Architecture (MDA). n. June, p. 1–15, 2003.

_____. OMG Unified Modeling Language Specification (draft). Versão 1.4. Disponível em: <http://goo.gl/1fiwNT>. Acesso em 14/04/2015. 2011.

_____. Metadata Integration using UML, MOF and XMI. OMG and Tutorial Contributors: EDS, IBM, Enea Data, InLine Software, IntelliCorp, Kabira Technologies, Klasse Objecten, ObjectTime Ltd., Rational Software, Unisys. 2000. Disponível em <http://goo.gl/CYQCHw>. Acesso em 14/04/2015. 2000.

PAULO, U. D. S. et al. Especificação de requisitos: uma introdução. p. 1–26, 1996.

PFLEEGER, S. L. Engenharia de Software: Teoria e Prática, Prentice Hall do Brasil, 2ª Edição, 2004

POHL, D. Adapting Traceability Environments to Project-Specific Needs. Communications of the ACM, 41 (12), p. 54-62, 1998.

PRESSMAN, R. S.; MAXIM, B. R. Engenharia de software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

_____. Software Engineering: A pratictioner Approach. Editora: McGrawHill, 6ª Edição, Porto Alegre, 2010.

RAMALHO, F. PLP – Paradigma de programação orientado a modelos. Universidade Federal de Campina Grande. Disponível em: <http://goo.gl/h1a5DE>. Acesso em 14/04/2015. 2010.

RAMESH, B.; JARKE, M. Toward Reference Models for Requirement Traceability. IEEE Transactions on Software Engineering, vol 27, no. 1. p. 58-92. 2001.

RODRIGUES, E. Por que o PMBOK não é uma metodologia? Disponível em: < http://goo.gl/htGHRG>, 2007. Acesso em 15/12/2015.

SANCHES, L; SOSSOLOTE, R.; SOUZA, A. Aplicação da Engenharia de Requisitos para a compreensão de domínio de problema para sistema de controle comercial. 2014. Disponível em: <http://goo.gl/EWYnax>. Acesso em 15/12/2015.

SANCHEZ, P.; ALONSO, D.; ROSIQUE, F.; ALVAREZ, B. Introducing safety requirements traceability support in model-driven development of robotic applications. IEEE Transactions on Computers, v. 60, n. 8, p. 1059–1071, 2011.

SAYÃO, M.; CESAR, J.; LEITE, P. Rastreabilidade de Requisitos. Revista de Informática Teórica e Aplicada, v. 12, p. 1–30, 2005a.

SAYÃO, M.; CESAR, J.; LEITE, P. Rastreabilidade de Requisitos. Revista de Informática Teórica e Aplicada, v. 12, p. 1–30, 2005b.

SELIC, B.; The Pragmatics of Model-Driven Development. Publicado em Jornal, IEEE Software, Volume 20, Issue 5, Setembro, p. 19-25, 2003.
SEIBEL, A.; NEUMANN, S.; GIESE, H. Dynamic hierarchical mega models: Comprehensive traceability and its efficient maintenance. Software and Systems Modeling, v. 9, n. 4, p. 493–528, 2010.

SHARMA, R.; SOOD, M. A ModelDriven Approach to Cloud SaaS Interoperability. International Journal of Computer Applications, v. 30, n. 8, p. 1–8, 2011.

SIEGERT, E.; MARQUES, M. R. S.; BRISOLARA, L. Supporting model-driven requirements management in the embedded systems domain. Proceedings - 2013 Symposium on Computing and Automation for Offshore Shipbuilding, NAVCOMP 2013, p. 34–39, 2013.

SILVA, L.; LEITE, J.; BREITMAN, K. C&L: Uma Ferramenta de Apoio à Engenharia de Requisitos. Publicado em PUC-RioInf.MCC24. 2004.

SILVA, A. Model-driven engineering: A survey supported by the unified conceptual model. Publicado em Computer Languages, System e Structures. Volume 43, Outubro, 2015. p. 139-155. 2015.

SMITH, R. S. Writing a Requirements Document. p. 1–6, 2015.

SOMMERVILLE, I. Requisitos do usuário e do sistema e do software. 2004. Addison-Wesley, 6a. edição

_____. Engenharia de software. 8ª ed. São Paulo: Pearson Addison-Wesley, 2007.

SOMMERVILLE, I.; SAWYER, P. Requirements Engineering – A Good Practice Guide.1997.

SOUSA, K.; MENDONÇA, H.; VANDERDONCKT, J. Addressing the Impact of Business Process Changes on Software User Interfaces. Business-driven IT Management, 2008. BDIM 2008. 3rd IEEE/IFIP International Workshop on, p. 11–20, 2008.

SOTILLE, M. Diferenciando Requisitos, Restrições e Premissas. 2012. Disponível em: <http://goo.gl/UD6wfW>. Acesso em 15/04/2015.

SWITHINBANK, P.; CHESSELL, M.; GARDNER, T.; GRIFFIN, C.; MAN, J.; WYLLIE, H.; YUSUF, L.; Front cover Patterns : Model-Driven Development Using IBM. Contract, p. 252, 2005a.

SZABO, C.; CHEN, Y. A model-driven approach for ensuring change traceability and multi-model consistency. Proceedings of the Australian Software Engineering Conference, ASWEC, p. 127–136, 2013.

TANKOVIC, N. Model Driven Development Approaches: Comparison and Opportunities. Disponível em: < https://goo.gl/Q8QGn6 >. Acesso em 04/04/2015. 2014.

TANKOVIC, N.; VUKOTIC, D.; ZAGAR, M. Rethinking Model Driven Development : Analysis and Opportunities. Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference, p. 505–510, 2012.

TORANZO, M.; CASTRO J.; MELLO, E. "Uma proposta para melhorar o rastreamento de requisitos". Em: WER03 – *WorkShop* em Engenharia de Requisitos, Valencia, Espanha. 2002.

TURINE, M; MASIERO, P. Especificação de Requisitos: Uma Introdução. 1996. Disponível em: http://goo.gl/U92YYz>. Acesso em 15/04/2015.

VALDERAS, P.; PELECHANO, V. Introducing requirements traceability support in model-driven development of web applications. Information and Software Technology, v. 51, p. 749–768, 2009.

VENTURA, M. M. Pedagogia Médica O Estudo de Caso como Modalidade de Pesquisa The Case Study as a Research Mode. Rev SOCERJ, v. 20, n. 5, p. 383–386, 2007.

XAVIER, R.; DE OLIVEIRA, A. A; DO NASCIMENTO, R. P. C. ModelER: Abordagem Baseada Em Modelos Aplicada Ao Processo De Elicitação De Requisitos. Proceedings of the 7th Euro American Conference on Telematics and Information Systems, p. 11:1–11:7, 2014.

WANG, J. W. J.; LIU, X. L. X. A Task-Based Modeling Method for Process Modeling and Automation in Project Management. 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, p. 8–11, 2008.

WESTFALL, L. Software Requirements Engineering: What, Why, Who, When, and How By Linda Westfall. ASQs Software Quality Professional Journal, n. 2004, p. 9–15, 2006.

WIEGERS, K. Software requirements. CS2 Software Engineering note 2, v. 3, p. 1–9, 2004.

WINKLER, S.; VON PILGRIM, J. A survey of traceability in requirements engineering and model-driven development. Software and Systems Modeling, v. 9, p. 529–565, 2009.

WOODSIDE, M.; PETRIU, D.; FAISAL, A.; A Systematic Approach for Composing General Middleware Completions to Performance Models in Computer Performance Engineering. Proc. European Performance Engineering Workshop EPEW14, Florence. LNCS vol. 8721, Springer, p. 33-44, 2014.

ZAVE, P. Classification of research efforts in requirements engineering. Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95), v. 29, n. 4, 1995.

ZOHREVAND, Z.; BIBALAN, Y. M.; RAMSIN, R. Towards a framework for the application of Model-Driven Development in Situational Method Engineering. Proceedings - Asia-Pacific Software Engineering Conference, APSEC, p. 122–129, 2011.